A Computational Model for Generating and Analysing Architectural Layouts in Virtual Environments

Arash Bahrehmand

TESI DOCTORAL UPF / 2016

CO-DIRECTORS DE LA TESI
Dr. Josep Blat, Dr. Alun Evans

DEPARTMENT OF INFORMATION AND COMMUNICATION TECHNOLOGIES

Acknowledgements

First and foremost I would like to express my sincere gratitude to my advisors Prof. Josep Blat and Alun Evans for the continuous provision of my Ph.D research, for their useful comments, motivation and engagement through different stage of my project. They have been tireless and kind in helping to deal with difficult problems over the course of these four years.

My sincere thanks also goes to Dr Thomas Batard who helped me to solve mathematical problems in the thesis. Without his valuable support it would not be possible to conduct this research.

Last but not the least, I would like to thank my family for supporting me spiritually throughout writing this thesis and my life in general.

Abstract

Designing interior layouts is one of the most common elements of the immersive computer graphics projects (e.g., games, virtual reality and special effects). The design of 3D virtual environments aims to provide not only a visually engaging experience, but also a plausible understanding of the virtual entities through which the user can easily associate real world objects with corresponding 3D digital models. Nowadays, digital artists apply Computer Aid Design (CAD) techniques to draft floor layouts. The planning process, as a complex human activity, becomes more prone to error when the designer is faced with different levels of uncertainty in a multidimensional problem such as this one. Despite the growth of computerized computational methods to generate and simulate floor plans, mostly based on Artificial Intelligence techniques, such methods are not satisfactory enough among designers yet, due to the weak problem formulation of the designers practices and that current methods do not incorporate subjective aspects of the design in the optimization processes.

This thesis contributes in two main fields of floorplan layout computation: computational quality metrics and procedural generation of 3D layouts. We introduce novel metrics based on formulating architectural standards, to measure the quality of *privacy* (as complementarity of visibility) and of *circulation*. We introduce two different approaches to generate floorplans, one based on subdivision, and focused on enhancing circulation, and another one based on hybrid optimization methods, supporting a wide variety of quality measures, and subjective input. The hybrid optimization algorithm takes advantage of an evolutionary strategy to generate a set of optimal solutions. In order to reach higher quality offspring at each generation and faster convergence towards optimal solutions, a parent selection method is proposed that attempts to find the most appropriate sub layouts in the recombination process (i.e., sub-layouts that more likely generate higher quality children after recombination). In addition, the subjective and contextual aspects of the design are addressed by incorporating user opinion in the fitness function of the optimization algorithm.

Resumen

El diseño de las disposiciones interiores es uno de los elementos más comunes de los proyectos de gráficos por ordenador y de inmersión (por ejemplo, juegos, realidad virtual y efectos especiales). El diseño de los entornos virtuales 3D tiene como objetivo proporcionar no sólo una experiencia visualmente atractiva, sino también una comprensión plausible de las entidades virtuales a través de la cual el usuario puede asociar fácilmente objetos del mundo real con los correspondientes modelos digitales 3D. Hoy en día, los artistas digitales aplican técnicas de diseño asistido por ordenador (CAD o Computer Aided Design) para elaborar diseños de planta. El proceso de planificación, como toda actividad humana compleja, se vuelve más propenso a error cuando el diseñador se enfrenta a diferentes niveles de incertidumbre en un problema multidimensional como éste. A pesar del crecimiento de métodos computacionales computerizados para generar y simular los planos de planta, en su mayoría basados en técnicas de inteligencia artificial, sin embargo, tales métodos no son lo suficientemente satisfactorios para los diseñadores, debido a la formulación débil del problema de las prácticas de los diseñadores y a que los métodos actuales no incorporan aspectos subjetivos del diseño en los procesos de optimización.

Esta tesis contribuye en dos campos principales del diseño computacional de plano: en las métricas computacionales de calidad y en la generación procedimental de las disposiciones 3D. Se introducen nuevos parámetros para medir la calidad de *privacidad* (como complementariedad de visibilidad) y de *la circulación*, basados en la formulación computacional de estándares de arquitectura. Por otra parte, se introducen dos enfoques diferentes para generar planos computacionalmente, uno basado en la subdivisión, y centrado en la mejora de la circulación, y el otro basada en métodos de optimización híbridos, que admite una amplia variedad de medidas de calidad, y de entradas subjetivas. El algoritmo de optimización híbrida aprovecha una estrategia evolutiva para generar un conjunto de soluciones óptimas. Con el fin de llegar a la descendencia de mayor calidad en cada generación y una convergencia más rápida hacia soluciones óptimas, se propone un método de selección de los padres para intentar encontrar los sub-diseños más adecuadas en el proceso de recombinación (es decir, sub-diseños que con más probabilidad generan los descendientes de mayor calidad después de recombinación). Además, los aspectos subjetivos y contextuales del diseño se tratan mediante la incorporación de la opinión de usuario en la función de aptitud del algoritmo de optimización.

Preface

In the past couple of decades, we have been witnessing the transition from Computer Aided Design (CAD) to Computational Design. This means the emergence of a whole new set of design concepts in both representation and solution of architectural problems. This novel design approach applies principles from different domains of Artificial Intelligence, Mathematics, Evolutionary Biology and Philosophy, among others. Artificial Intelligence based approaches have shown a considerable potential to address multiple facets of design problems. The generated results of computerized design tools should be measured by some computational metrics as well.

This dissertation is an original intellectual product of the author, Arash Bahrehmand in the field of computational design and optimization, specifically for the issue of providing floor layouts. The background of the work is presented in Chapter 1, two computational metrics are presented in Chapter 2 and 3, while Chapter 4 and 5 present two layout optimization methods. Among these chapters, Chapter 5, presents a more ambitious optimization method which takes advantage of an evolutionary strategy and supports irregular polygons to reach a set of optimal solutions. The proposed metrics and optimization methods have been assessed through different case studies.

Contents

List of Figures

List of Tables

Chapter 1

INTRODUCTION

1.1 Motivation and Problem Statement

Computational design is one of the most common tasks of immersive computer graphics projects, such as games, virtual reality and special effects [Lobos and Donath, 2010, Indraprastha and Shinozaki, 2012a]. In this work, we consider two types of (computational) designers: *architects* and *digital artists*. An architect is a professional who is qualified to design a building based on aesthetical and functional requirements. A digital artist is a designer who applies graphic design principles and his/her creative skills to generate virtual content for video games or visual effects.

This thesis is focused on *layout planning*, which is a significant subcategory of computational design, and recently has been the interest of research in various layout configuration problems, such as texture atlases, building arrangement in urban design, furniture arrangement, and interior design [Peng et al., 2014], which we outline next.

Designers take advantage of layout planning techniques in *arranging space units of a building* (either virtual or real), and this arrangement has been studied by several researchers [Merrell et al., 2010, Rodrigues et al., 2013a, Peng et al., 2014], and, similarly, closely related methods are applied to *arrange furniture* in a given layout [Yu et al., 2011, Merrell et al., 2011]. We discuss more deeply layout planning next.

Besides from architectural applications of the layout planing, in real time computer graphics, several automatic layout solvers are proposed to arrange sub-images in a larger image, called texture atlas [Guthe and Klein, 2003]. A *texture atlas* is a large image container in which sub-images are collected, each of them being a texture for a particular part of a 3D or 2D model.

Currently, architects draft floor layouts applying Computer Aid Design (CAD) techniques. The planning process, as any complex human activity, becomes more prone to error when the designer is faced with different levels of uncertainty in multidimensional problems. Indeed, despite the growth of computational methods to generate and simulate floor plans, such methods are not quite satisfactory among designers yet. Our hypothesis is that this unsuitability of the solutions so far comes, on one side, from the fact that the formulation of the architect's practices is weak

and also that the methods could be improved by incorporating subjective aspects of the design in the processes.

Traditionally, early layout planning starts by drawing design drafts wherein the designer applies an extensive manual process of trial-and-error to evolve the quality of schematic designs based on his past experiences and basic rules of thumb [Bhatt et al., 2013]. In general, most of architectural projects share the following characteristics in their design process:

- The final layout highly depends on the manual heuristics that are applied in drawing the first drafts. Therefore if the first schematic designs suffer from lack of some desired qualities, then there is no guarantee that an architect will be able to improve the quality of the design in the following iterations

- The process of resolving the design issues with the intention of evolving the current state of the design is far from being straightforward, due to the contradictory nature of the problem constraints in architecture design.

- Relying only upon personal past experiences of architectural projects is not always sufficient to reach a solution, since each project may emphasise on a new aspect of the design and may require a new strategy to fulfill the constraints, one that has not been addressed in the previous projects of the designer. And if s/he has worked on a limited number of projects in the past, the process of finding a project that has some level of similarity to the current project is not clear.

Thus, the process leading to the layout plans by architects becomes difficult and potentially with inaccurate results for projects with complex and contradictory constraints. These are reasons behind the growing research towards computational approaches to this problem, which would ease the task of the architects, and lead to solutions whose accuracy can be better assessed.

On the other hand, the demand for automatic design and creation of digital content to be used in virtual environments, has been growing dramatically in the past few years [Ritchie et al., 2015]. Interiors and buildings form the largest part of the virtual content to be created by digital artists [Leblanc et al., 2011]. The design of 3D virtual buildings aims to provide both a visually engaging experience [Ma et al., 2014] as well as a plausible understanding of the virtual entities so that the user can easily associate real world objects with corresponding 3D digital models [Vasin et al., 2015]. However, creation of virtual layouts can be even more complex and time consuming for digital artists than for architects. First, the former do not usually have enough knowledge to address architectural requirements of the

building; and second, unlike architects who normally concentrate on a single layout at a time, digital artists may need to create a mass of buildings for a particular district of a virtual city.

AI-based CAD tools have shown a potential to address the issues mentioned above by providing alternative design solutions, measuring the quality of each alternative and proposing design ideas based on data-driven approaches. However, despite the growth of computer tools to generate and simulate floor plans, they are not sufficiently satisfactory among designers yet [Rodrigues et al., 2013a]. We believe this is due to a number of reasons, and among them the following ones are relevant:

- **Weak problem formulation of the designers' practices.** The tools so far can only generate a particular type of building layouts. For instance, as we discuss in Chapter 4 and 5 most of them only produce rectangular layouts while architects may need to configure or arrange irregular shape spaces in a layout problem.

- **Not providing user-friendly interfaces.** Most automatic layout solvers remain in a research stage and have not been evaluated by many digital artists. The provided CAD tools are far away from the type of user interfaces that the digital artists use to work with [Lobos and Donath, 2010].

- **Not incorporating subjective aspects of the design in optimization process.** Although layout solvers are able to fulfill the formal functional requirements of the project, they normally fail to fully satisfy the aesthetic aspects of the layout due to the contextual and subjective nature of the design [Xu et al., 2015, Parish and Müller, 2001, Coates et al., 2005].

An appropriate computerized layout solver should provide an accurate formulation of the architectural problems; should be based on a robust reasoning system of the architectural qualities, both functional and subjective; and incorporate friendlier user interfaces in order to support better the process of automatic layout creation.

1.2 Research Questions

One of the main practices in architectural design is predicting the influence of the built elements on peoples experiences [Key et al., 2008, Wiener and Franz, 2005]. Design in architecture is an iterative process, and appropriate evaluation of the layout properties increases the accuracy of the process. The improvement of the design quality at each iteration relies on decision making processes which are performed on the previous results. Moreover, the designer can use the measure of the

architectural quality of a layout to identify which aspects of the design need more adjustments. However, measuring the quality is a difficult task in multidimensional and complex projects.

Indeed, the computational representation of space attributes might lead architects to a better understanding of the spatial concepts [Key et al., 2008], resulting in a more accurate reasoning in the iterative process of finding solutions. Therefore, an accurate quantitative representation of the spatial environment can support the architect in estimating the consequences of each decision. The efficiency of the evaluation mechanism is an important aspect, since the architect needs to assess the quality on the various designs at each iteration step. Hereby, computational metrics can be a practical alternative to traditional evaluation methods if the architect experiences a tangible acceleration in the design process, if their quality is appropriate.

Thus, the first main research question of this thesis is **How can architectural practices or rules be formulated into computational metrics that provide accurate and efficient assessment of a layout?**. In the chapters 2 and 3 we attempt to address it by discussing the feasibility, in terms of accuracy and efficiency of two quality metrics that we propose, for *circulation* and *privacy*. In addition, in chapter 5 we propose three other quality metrics, namely: *Topological*, *Overflow* and *Compactness*.

In the early stages of the design, the architect may start to work by selecting the basic theme of the initial layout among layouts that have similar characteristics to the current project resulting from his/her previous work. Due to the limited number of layouts that can be reviewed by a person before s/he starts to draft the design ideas, many potential layout solutions that may have better quality are not considered in the initial reviewing process. Computational generative tools can benefit the designer by providing various layouts which satisfy a set of constraints. However, only generating a large number of layouts is not so helpful, since finding an appropriate layout among many layouts would be another demanding task for the architect. Therefore, the computerized methods should be able to provide a huge space of layout solutions, as well as to find the optimal layouts.

Appropriate optimization methods should take into account the architectural functionalities of the design, as well as the aesthetic aspects of the layout [Karlen, 2011], efficiently when measuring quality factors and generating optimal solutions. The evolutionary computing methods generate a random set of solutions and attempt to improve the quality of the random layouts in an iterative manner. Due to the contextual nature of the design, defining a computational formulation of the human perception about *beauty* is not intuitive [Wiener and Franz, 2005]. Indeed, the beauty of an architectural design varies from one culture to another depending

upon conceptual factors and historical attributes of the site where the building is going to be built. Furthermore, the perceived quality of a design or a layout can be different for each individual, even within the same cultural context, due to the variation of personal characteristics and social background [Demirbas and Demirkan, 2000]. Therefore, incorporating the designers opinion in the optimization method becomes essential with the goal of personalizing the layout solutions.

In order to handle complex design problems, most of the optimization algorithms attempt to generate valid solutions by formulating the problem requirements into a set of *hard* and *soft* computational constraints. *Hard* and *soft* constraints are explained in more detail in chapter 5. Although determining design constraints can guide the optimization methods to look for optimal solutions efficiently, having a huge number of constraints decrease the chance of reaching creative solutions.

Therefore, the second question of this research is **How can an optimization method be defined so that it generates and finds the optimal layout solutions based on architectural guidelines and user's preferences?**. This question is addressed in chapters 4 and 5 by proposing two optimization methods, a binary tree based one and an evolutionary computing one, respectively.

1.3 Methodology and Contributions

Thus, in this thesis optimization of 3D layouts and computational quality metrics are the key subjects of research, among other relevant techniques, which were considered. Several types of optimization techniques, such as data driven approaches [Merrell et al., 2010], simulated annealing [Yu et al., 2011] and genetic algorithms[Koenig and Knecht, 2014], have been applied in layout computations. In interactive computer graphics, layout generative tools have used by digital artists to decrease the complexity of the design in creating virtual buildings [Müller et al., 2006], furniture arrangements[Fisher et al., 2011] and *texture atlas*, large image containers in which sub-images are collected, each of them being a texture for a particular part of a 3D or 2D model. Texture atlases have been used in real time computer graphics to optimize the rendering cost [Lévy et al., 2002, Sander et al., 2001], as efficient automatic arrangement of the sub-images layout is very important [Guthe and Klein, 2003]. Most automatic content generative methods have proposed and/or applied some computational metrics to evaluate the quality of generated contents, for example, circulation [Bahrehmand et al., 2014a], visual openness and privacy [Indraprastha and Shinozaki, 2012a].

On the other hand, a framework has been implemented to assess the applicability of proposed approaches. The framework is interactive, which is required by the subjective aspect of the design process, as it has been previously discussed. The

framework involves two main types of elements, related to quality analysis component and to procedural generation, respectively. The quality analysis elements measure the quality of layouts based on some computational architectural metrics, such as circulation, privacy or harmony. The procedural generation elements are oriented towards creating 3D layouts taking into account architectural guidelines and user's preferences.

The evaluation of the results obtained has been carried out along different axes: the quantitative performance, the competence to support users to achieve layouts with some overall predefined characteristics or constraints, its validation with respect to producing human made plans, the comparison of features with relevant systems, the characteristics of the algorithmic approaches, among others. The quantitative analysis is conducted by assessing different aspects of the algorithm by running several experiments over a set of scenarios in which each scenario represents a unique case study (see Chapter 5 ref...). The results demonstrate the scaleability and robustness of the proposed methods while the characteristics of the final layouts are close to user preferences. The validation tests are designed to measure the similarity of the artificial (generated) layouts to the reference floor plans that are designed by human architects. In (Ref later) two validation tests are presented to evaluate subdivision based and EA based optimization approaches, respectively. In both experiments the presented methods managed to generate layouts that are roughly same (similar to) as the reference floor plans. In some cases, the perception of specialized users has been included. For instance, in Chapter 2 the effectiveness of the presented circulation metric is measured through an informal study with five architect participants.

1.3.1 Computational Quality Metrics

As mentioned above, architects need to assess the quality of the space, usually to understand up to which extent it fulfills some constraints. This spatial quality is usually evaluated in terms of the geometrical attributes and the topological relationships to other spatial units. In this work, the major contributions with respect to computational metrics are:

- **Providing a computational solution to analyze, visualize and evaluate the circulation in 3D layouts.** The circulation quality measure that we propose takes into account several attributes for each path in the layout: length, complexity and desired topological constraints. A *path* is defined as connecting the centers of two space units to each other satisfying the layout restrictions. The input of the proposed algorithm to measure the circulation quality consists of the 3D geometry of the layout and the desired topological

constraints set by the user, in the form of a matrix. The main advantage of the proposed metric is to provide a more accurate evaluation of circulation by involving factors that have been neglected in relevant work. For instance, in order to measure how close a layout is to high level requirements of the project, the topological graph is automatically extracted from the 3D model of the layout and compared to the desired topological relationships. A discussion of the circulation metric is contained in chapter 2.

- **Providing a computational solution to analyze, visualize and evaluate the privacy/visibility of 3D layouts.** The proposed method measures the privacy of each space unit based on computing the visibility of all possible points of view within the 3D geometry of the space unit. The proposed algorithm input is the 3D geometry of the layout along with the transparency level of architectural elements. The main contributions of this approach are measuring the visual privacy from all points of the space units (instead of only the center point) and involving the importance level of *being viewed from a particular space unit* in privacy calculation. A discussion of the privacy metric is contained in chapter 3.

To support better the designer's understanding of circulation and privacy, a 3D visualizer that illustrates circulation paths and privacy level of different points inside and outside a layout is provided.

Moreover, both the circulation and privacy metrics are used in our optimization algorithms to measure the quality of the layouts generated at each step of our solver, along other computational metrics (topological, overflow, and compactness), whose concepts are inspired by relevant works [Rodrigues et al., 2013a, Afyouni et al., 2012]. The topological quality metric assesses how close the adjacency matrix of the generated layout is to what the user specified at the initial stage as the desired topological constraints. The overflow metric is computed based on the fitness of a layout contour to the site boundary while the compactness quality determines the appropriateness of a layout in terms of removing useless gaps between space units.

1.3.2 Layout Solvers

In terms of actual solvers, providing layout solutions automatically, the thesis presents two novel approaches:

- **An Automatic Subdivision-Based Layout Solver.** Two criteria that have to be usually taken into account by automatic layout solvers are: absence of space units overlap and maximizing the compactness. Subdivision based

methods have been recognized as an appropriate solution to address the requirements mentioned. However, the methods proposed so far suffer from a main drawback, which is the poor support of non-rectangular shapes. We have proposed a BSP tree construction method that splits a given space through a top-down approach and trace the generated sub trees through a down-top or bottom-up approach to filter inappropriate solutions. On the other hand, we have introduced and applied the concept of manual and random null space assignment to support non-rectangularity in subdivision based methods. A discussion of the proposed EA based optimization system is contained in chapter 4.

- **An Interactive Arbitrary Shape Layout Solver.** To the best of our knowledge most of the computerized layout solvers so far have two main drawbacks: they do not support subjective aspects of the design, which leads to user insatisfaction; and they do not support irregularly shaped polygons. To tackle these problems we propose an evolutionary computing based solver which:

  - accepts user preferences which are turned into evaluation parameters of the layout at each iteration. In fact, we propose a similarity metric comparing new generated solutions to what the user specified as his/her favorite layouts in previous generations. Thus, the user opinion contributes to personalize the layout solutions based on his/her preferences.

  - generates layouts supporting arbitrary polygon shapes for both input space units and layout contour.

  The introduced method is based on evolutionary computing techniques; the input consists of high-level dimensional and topological constraints and the output is a set of optimal layouts. The fitness function of the evolutionary algorithm takes into account several spatial quality metrics, as discussed above and user preferences (through the similarity metric). The proposed evolutionary algorithm applies a multi-parental recombination operator to improve the chance of generating high quality offspring that results in faster convergence of optimal solutions. The system has other user oriented features as well, as outlined in the following subsection. A full discussion of the proposed evolutionary computing based optimization strategy is contained in chapter 5.

1.3.3 A User Oriented Implementation

According to [Lobos and Donath, 2010, Indraprastha and Shinozaki, 2012a] designing interiors is one of the most common tasks of immersive computer graphics projects, such as games, virtual reality and special effects. Our generative tool based on evolutionary computing has the relevant characteristic of immediacy, which improves the reasoning quality in the design process, as an appropriate visualization of 3D interiors provides the designer with a better understanding of the current stage of its design. On the other hand, our application provides the user with an interactive interface to specify constraints, visualize the generated layout solutions and choose some of them to express his/her preferences. The software application is able to illustrate the set of all possible paths between space units (see Chapter 2), visualize by color all possible points of the layout based on visual privacy value (see Chapter3), draw 2D and 3D layouts, and also it allows the user to add fine tune modifications to the final layouts through removing/adding openings and walls (see chapter 4 and 5).

1.4 Thesis Outline

This thesis is structured in seven chapters. Chapter 2 and 3 present computational models to measure the quality of circulation and privacy, respectively. Chapter 4 and 5 are dedicated to procedural generation of layouts, while in Chapter 4 a subdivision-based layout solver is presented and Chapter 5 introduces an interactive layout generator which applies genetic optimization algorithm to find optimal solutions. In Chapter 6 the implemented software application is demonstrated and discussed from the software engineering perspective. Overall conclusions and future work are presented in the last chapter of this thesis.

Chapter 2

MEASURING THE CIRCULATION QUALITY

Title	A Computational Metric of the Quality of Circulation in Interior Spaces
Authors	Arash Bahrehmand, Alun Evans, and Josep Blat
Conference	Information Visualization Theory and Applications (IVAPP), 2014 International Conference on
Year	2014
Keywords	Space Quality, Circulation, Visualization, Visibility, Computational metric, Graph Based Grid, Path Finding.

A Computational Metric of the Quality of Circulation in Interior Spaces

2.1 Abstract

Space, in terms of interior and exterior design, is one of the most important issues facing all architects. In particular the movement of people through sequences of spaces forms a large part of the circulation problem in architecture planning. Although several studies have applied network models on urban analysis to take advantage of graph based queries, understanding interior design principles based on graph attributes shows potential for further research. This paper presents a computational solution to analyse, visualize, and evaluate the circulation quality of indoor spaces. To achieve it, first we create a grid graph based on a geometrical representation of space. Using this grid, a semantic weighted graph is generated, that helps us to provide a measured score for the circulation of people in a given space. The results were tested against architects scoring, showing that the measure is adequate. We also discuss the efficiency of our approach.

2.2 Introduction

Over the past few years, scientists have been applying advances in fields such as Artificial Intelligence or Computer Graphics to address multifaceted problems through intelligent applications. This is part of a growing digital revolution that has been dramatically transforming traditional disciplines. Architecture is among the most prevalent fields, and has received considerable attention from researchers, with the aim of improving the design phase and visualizing architects ideas. Recently, researchers are focusing on a new trend of design methods that exploit computational approaches to measure the quality of design elements (e.g. windows, columns, beams) from various points of view. However, there is still a need to provide strong support for architects creativity through computerized methods, which assess the space quality.

Space, in terms of interior and exterior design, is one of the most important issues facing all architects. In this context, it can be defined as a collection of connected points satisfying particular geometric constraints. Judging the quality of a space means assessing to which extent the space configuration satisfies the expectations of the designer and the client. Spatial measurement solutions help an architect to evaluate how near each of his/her different plans are to the project objectives. A wide range of methods can be used to measure quality of space plans to obtain an

appropriate view of their consequential spatial quality, before a final decision about the plan to be implemented is made. For instance, accurate statistical information could help to analyse how the configuration of architectural elements influences people's experience and behaviour. This is especially the case for large projects, involving numerous objectives, where an architect needs improved analysis tools. This analysis entails a creative consideration of all quality factors, where there is a need to determine the programmatic principals in a physical arrangement to satisfy the clients demands.
Movement patterns of people can be influenced by the perceptual thread that connects different points of the built spaces. Circulation is a substantial element in interior design, and architects early designs include a relationship matrix that defines the essence of the accessibility among rooms. To support the transition from this matrix to a more creative space planning, an analytic tool of circulation that takes into account different principles of design will be needed. The principal contribution of this paper is a computational solution to analyse, visualize, and evaluate the circulation quality of indoor spaces, providing circulation scores to 3D plans, in order to help architects to decide among different designs. Our approach accepts a 3D plan and a relationship matrix as inputs. Then an algorithm extracts a grid graph at a fine level of granularity that contains all the geometrical properties of the plan. In the next step, a topological graph is generated that reveals the cost of movement among different spaces, and the traffic flow cost of a 3D plan is calculated. Finally, circulation quality is measured based on similarity of the topological graph and the relationship matrix, and the traffic score of the given 3D plan.
The remainder of the paper is organized as follows. Section 2 provides a review of some related work that attempted to measure circulation quality in architecture. Section 3 introduces our proposed approach. Section 4 discusses the experimental results for three different 3D plan. Finally, a conclusion and discussion based on our finding from this study are presented in section 5.

2.3 Related Work

One of the earliest studies in the field uses a shared concept between architecture and geography, *isovist* [Benedikt, 1979], which is defined as the part of space visible from a given vantage point. The vantage point is the position of the viewer so that the quality is measured based on his/her point of view. Thus, isovist is a smart way of understanding an interior environment from the point of view of individuals, as they interact with it. This obtained visible space is associated with different measures such as area, distance, and occlusion. [Kelly and McCabe,

2006] takes the advantages of isovist in path finding algorithms through a visibility graph. [Wiener and Franz, 2005] try to find out a relationship between spatial characteristics of buildings and spatial experience and behaviour of people.

Architecture is not a static experience but is experienced dynamically through circulation in the space[Puusepp, 2011]. Church [Church and Marston, 2003] introduced a comparative access measurement that can be combined with traditional measures of absolute access to assist architects in making decision about finding optimized paths in urban design. Paul C Merrell et al. presented an intelligent approach for generating residential building layouts automatically [Merrell et al., 2010]. Their method takes advantages of machine learning and optimization techniques for producing plausible building layouts. Although in the optimization procedure the accessibility term, along with other architectural terms, is applied for cost evaluation, it only considers the number of missing connections and entrances. Building Information Modelling (BIM) is the process of producing and managing data involving digital representations of physical and functional characteristics of a building during its life cycle. [Lee et al., 2008] present a BIM-enabled graph application for analysing accessible routes within indoor spaces. They use an accessible distance measurement technique and provide a visualization system highlighting spaces that are in the path. In the field of interior spaces, much work has been done to provide a spatial model for measuring the navigations quality between different space units. In addition, some studies concentrated on location-aware navigation in the form of navigation queries that help the users to find a point of interest through evaluating some factors such as travel time [Afyouni et al., 2012]. According to [Afyouni et al., 2012] two types of spatial models are recognized: geometric and symbolic spatial models.

2.3.1 Geometrical Representation

Geometric spatial models are based on geometrical characteristics of the space. A widespread approach in the field consists of splitting the plan into certain number of non-overlapping parts. A well-known gridbased approach uses a regular tessellation method. Moravec et al [Moravec and Elfes, 1985]present highresolution spatial maps in a system that navigates a mobile robot to a desired destination. Although grid based approaches are appropriate for navigation and easy to implement, they are expensive in terms of memory and processing time for large spaces. This well-known geometric structure splits a space into regions close to a set of particular points of interest [Choset, 1997].The main drawback of Voronoi tessellations is that, in some situations, the path may not be optimal [Afyouni et al., 2012].

2.3.2 Symbolic-Based Models

Symbolic-based approaches try to generate a graph based on topological characteristics of a given space [Dürr and Rothermel, 2003], where nodes are semantic locations (e.g., rooms, doors) and edges are connections that provide the possibility of movement between locations [Choset and Burdick, 2000, Remolina et al., 1999]. Place based graphs are the general form of symbolic graphs where nodes are rooms and edges are doors connecting rooms. This modelling approach has been receiving much attention in navigation planning and answering nearest neighbour queries.

In [Dik-Lun et al., 2004]a semantic model is presented where the classic place-based approaches are associated with some more knowledge such as the distance between nodes. [Allison et al., 2010]define a grid graph-based model of an indoor plan. The space is divided into some spatial units according to the floor plan, and then these units are represented by a grid graph where nodes and edges are labelled based on their belonging to spatial units. Their modelling approach can be applied in route, diffusion, and topological analysis.
To sum up, the common limitations of the presented methods naturally fall into one of three categories: ignoring site-specific aspects; overlooking the purpose of the building when generating semantics behind the symbolic graph; and finally the lack of a combined approach that takes advantage of both grid and symbolic graph at the same time.

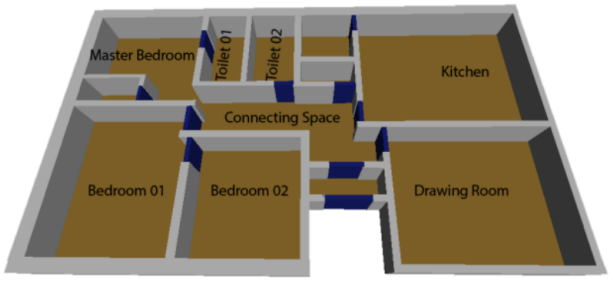


Figure 2.1: The input 3D plan

2.4 Proposed Approach

This paper addresses how an architect can select the best (plan) among different creative design alternatives in terms of circulation functionality. While there are

several guidelines for configuring architectural elements, the main motivation behind all of them is to design architectural spaces to be unobtrusive and efficient, so as to support all possible accessibility requirements. The matrix format is a commonly used method for organizing information in the pre-design stage. The density and complexity level of this matrix depends on the size and project requirements [Karlen, 2011]. As pointed out above, architects often use a special type of matrix, called relationship matrix, representing relationships and adjacencies between spaces. The relationship matrix consists solely of an interpretation of accessibility information and does not propose any planning solution. Therefore, in the design process architects should comply with the expectations set out in the relationship matrix. Finding the best design solution in large projects, with a dense matrix, is typically not interesting for the analyser, and it is prone to error. In order to have an accurate understanding of accessibilities in an environment, our algorithm accepts both relationship matrix as an input as well as 3D plan that is annotated by the architect. This is a key innovation of the method we propose. Figure 2.1 illustrates an annotated 3D floor plan. Annotations help us to identify the functionality of each sub-space in the building. We proposed a similarity metric that measures the similarity of a symbolic matrix of a given 3D plan to the relationship matrix. In addition, several factors that are not addressed by similarity measurement, e.g. traffic and overall travel cost, are taken into consideration in measuring the circulation quality of a 3D plan.

2.4.1 Creating the Grid Graph

As pointed out above, the grid-based model is a well-known approach for representing navigable and impassable regions in space by assigning different labels to graph nodes. In order to create automatically a fine grid-based graph based on the geometrical attributes of 3D plan, we use a ray casting method. The granularity of the graph depends upon the partitioning complexity of the plan. Graph nodes, called *GNodes*, represent predefined places that have been extracted automatically from geometrical structure in the 3D plan. Each node has a label, for symbolic graph extraction, and at maximum 8 neighbours for navigation purposes. First, a grid-based graph is created on top of the 3D building, according to the bounding projection of the 3D plan. Then, from each GNode a ray is cast down the 3D plan and, based on the collision of the ray and the 3D element inside the plan, the label of the corresponding node is determined. If the collision is detected on the wall the label is set to impassable, otherwise the label is assigned a value according to the spatial unit detected by a ray colliding with the building ground. As mentioned above, annotations reveal the name of each spatial unit in the 3D building, therefore these names are applied for determining label values of grid graph nodes. For

instance, if a ray collides with kitchen ground, the corresponding grid node gets the label value of kitchen.

2.4.2 Generating the Symbolic Graph

In this step, we use a grid base graph to generate a topological (symbolic) graph that presents the possibility and cost of moving from one space to another. Nodes, called *SNodes*, symbolize predefined space landmarks extracted from Gnode labels. Edges stand for the weighted connections that make it possible to interact between space units [Remolina et al., 1999, Werner et al., 2000, Remolina and Kuipers, 2004]. As pointed out above, GNodes are labelled according to their belonging to a corresponding subspace. In order to create SNodes, first, the GNodes are grouped based on their label values and then, according to each group, an SNode with a label corresponding to the inherited group label is created. The weight of each edge depends on the length and complexity of shortest path between two space units. Figure 3 illustrates a typical symbolic graph for the plan in Figure 2.2.

Shortest path distance

Shortest path is represented by an edge whose value is the length of shortest path, in terms of number of GNodes in the path, between the center of a space unit corresponding to center of other space. In order to find the shortest path an A* path finding on grid graph is implemented in a way that walls are considered as impassable objects. In order to normalize the shortest path distance, we divided it by the longest possible path distance in the floor. The longest path is a path that passes through all nodes in the grids without any duplication and ignoring impassable walls. The shortest path is calculated between two points that we calculate as the center points of the two corresponding spaces. Our definition of a center point is a point inside the space that has the minimum variation between its distances to all corner points of the space. The algorithm below describes the distance is calculated. In 2.1 the normalized value of shortest path is calculated.

$$NSP_{a,b} = \frac{SP_{a,b}}{SP_{max}} \tag{2.1}$$

Where $SP_{a,b}$ is the number of nodes in the path between a and b.

Path Complexity

It is generally accepted that people tend to walk along the easiest, simplest and most visible path [Lee et al., 2008]. Human navigation pattern relies on mental planning processes which are continuously updated based on individual current perceptual configuration of the space. In doing so, we measure the complexity level of a path based on substantial factors: path visibility and direction changing.

As pointed out above, isovist measures local spatial configurations in terms of visibility from a vantage observation point. Thus, each point in the space has a particular isovist value based on its position in the space. In Figure 2.4 an isovist map is illustrated based on the isovist value grid points where the brighter a point is, the more isovist value it has. Of course, the more a pedestrian knows about the configuration of the space though which his walking through it, the better s/he can find his/her way. Due to the isovist quantity, we can measure the perception level of an individual at each point of the path. Therefore, by summing up the isovist value of all points in a path we can assess the quality of view point along the path. In other words, the summation value determines the simplicity level of way-finding along a given path. In 2.2, NIsovisti is the normalized value of Isovist of GNodei and MaxIsovist is the maximum value of Isovist among all GNodes.

$$NIsovist_i = \frac{Isovist_i}{MaxIsovist} \tag{2.2}$$

Therefore in 2.3 $SPIsovist_{a,b}$ is the Isovist value of the shortest path between $GNode_a$ and $GNode_b$ and $SP_{a,b}$ is the number of GNodes in the path.

$$Iso_{a,b} = \frac{\sum NIsovist_{ii \in SP_{a,b}}}{MaxIsovist} \tag{2.3}$$

One of the most substantial factors that affects both simplicity and visibility is the number of directionchanges through the path. In this sense, one prefers to move in a path that is as straight as possible. Hence, the more the direction of the path is changed, the more complex the path is.

In order to recognize when the direction is changed we use a distance measurement hypothesis. A path consists of a series of connected nodes in a way that each node, except the first on, is connected to his parent node. In order to normalize the number of direction changing we have divide it by maximum possible number of direction changing in a path. In doing so, the maximum value happens when the

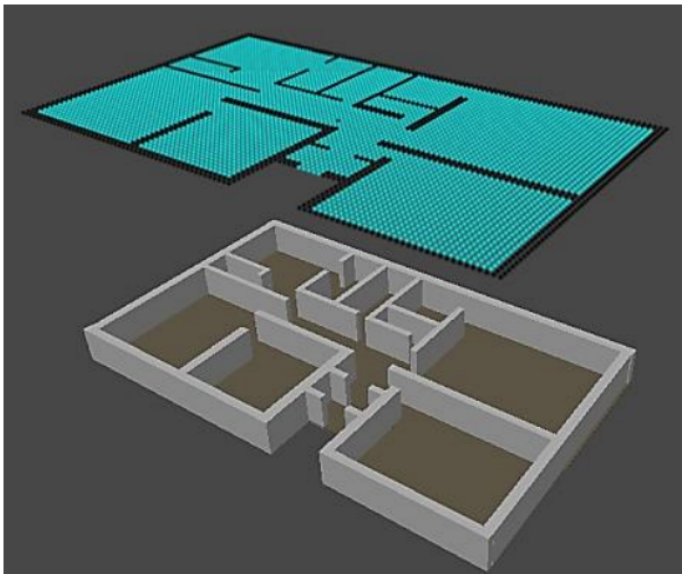


Figure 2.2: The input 3D plan

direction changes, approximately, in all GNodes.

$$NDC_{a,b} = \frac{NumOfDirCh_{a,b}}{|SP_{a,b}|} \tag{2.4}$$

Edge Weight

The weight ofthe edge between Snodes a and b is calculated through the combination of path complexity and shortest path distance of the path that connects space unit a to b. For example in a educational building, with many students and classes, finding a shortest path is substantial while in a museum the path length is not substantial but it should cover objectives of the expedition.

$$Weight_{a,b} = (1 - NPS_{a,b})^{\lambda 1} * (Iso_{a,b})^{\lambda 2} * (1 - NDC_{a,b})^{\lambda 3} \tag{2.5}$$

Where λi s adjust the weight between different terms based on the site-specific circumstances. For this λi paper, was kept at a value of 1.0

2.4.3 Calculating Similarity

Our similarity metric measures the similarity between two matrices: the relationship matrix and symbolic matrix.

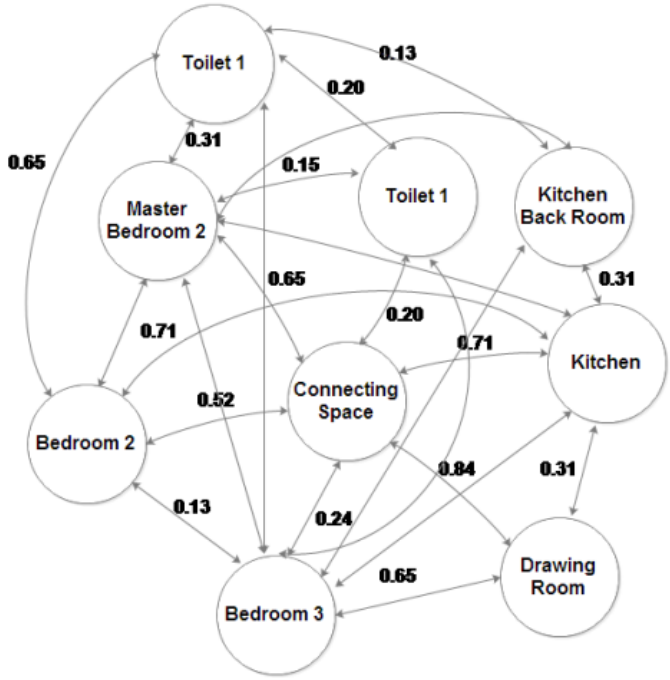


Figure 2.3: An example of symbolic graph of input plan.

The former is the input matrix that determines the accessibility type of space units, while the latter is the matrix representation of symbolic graph. In fact, symbolic matrix is an s by s matrix, where s is the number of space units. If there is a single door between space unit a to b, then the element $S_{a,b}$ is $Weight_{a,b}$, otherwise it is 0. The reason we used weights instead of binary representation of the matrix is because, even if two spaces are adjacent, the door position can still have a substantial influence on the circulation pattern.

On the other hand, the input relationship matrix (or adjacency matrix) represents three levels of connectivity importance, Must, Should and Could, for those space units that are connected through only one door. For instance, the importance level of those spaces that are connected by Must is much more important than those that are connected by Should. For the sake of using this matrix in similarity computations, instead of qualitative terms we use three equivalent quantitative values as 1, 0.5 and 0.25 for *Must*, *Should* and *Could* respectively. Table 2.1 and Table 2.2 Illustrate a sample convert from a relation matrix R to R'. Moreover, if a plan does not satisfy even one of the Must conventions, the plan should be ignored. In fact, the similarity determines how much the proposed plan satisfies relationship

matrixs conventions. The similarity of relationship matrix R and symbolic matrix S is calculated through 2.6. The more similarity, the more successful the proposed plan is in implementing relationship matrix demands.

$$PathSim_{R,S} = \frac{\sum R[i,j] * R[i,j]_{i,j \in R}}{|R|} \tag{2.6}$$

2.4.4 Traffic

In architecture, traffic is defined as the possible number of people who are walking in a space at the same time.

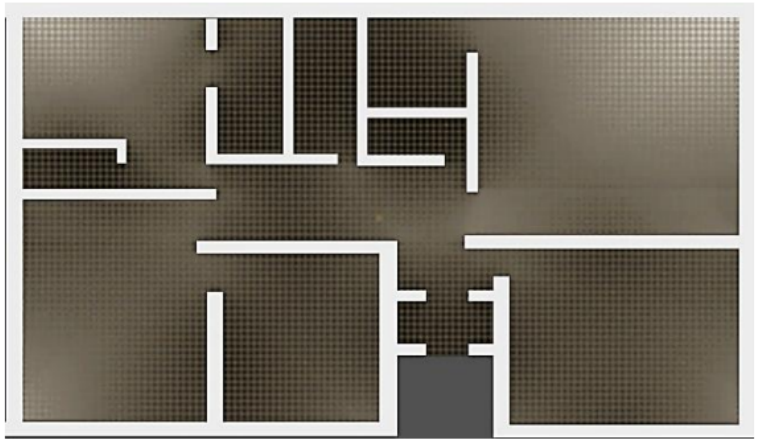


Figure 2.4: Isovist map of the floor plan.

In architecture design, a connecting space is understood as a particular space with disjoint address spaces and a set of links connecting pairs of space units and sharing the same channel [Araújo et al., 2009]. One of the most significant aspects of connecting space is the amount of possible traffic that may occur within this space. Although, increasing the size of connecting space can decrease the traffic, leaving a large space only for connecting space [Karlen, 2011]. Therefore, architects try to consider an appropriate size with lowest traffic for connecting space. In order to measure the traffic, first we should find the connecting space in the symbolic graph. The connecting space is the space that has most neighbours in the symbolic graph. In 2.7 quality of traffic for floor plan p is computed.

$$Traffic_p = \frac{Area_p}{Area_{ConnectingSpace}} * \frac{NumOfPath}{Area_{ConnectingSpace}} \tag{2.7}$$

Where $Area_{ConnectingSpace}$ is the number of Grid nodes in the connecting space.

2.4.5 Overall Path Efficiency

Overall path efficiency (OPE) calculates the summation of all possible shortest paths weights between all space units. The more summation of paths is the more efficiency can be realized for the plans circulation.

$$OPE = \frac{\sum Weight_{a,b_{a,b \in SpaceUnits}}}{\binom{|SpaceUnits|}{2}} \tag{2.8}$$

Where the SpaceUnits is the set of all space units in a 3D plan.

2.4.6 Circulation Quality

Finally, circulation quality is calculated through weighted combination of explicit and implicit factors. The relationship matrix is determined explicitly by architect while path complexity and overall path efficiency are inferred implicitly from the 3D plan. In 2.9 the circulation quality of plan *P* is measured and two parameters, α and β , are defined to adjust the weight of different factors where based on the plan application. These values are defined empirically and determine the significance of each factor in measuring the quality of circulation according to buildingt's practices and conditions. For instance, in hospital the significance of shortest path is much higher than other parameters, therefore the value of β should be increased.

$$CQ_p = \alpha * PathSim_{R,S} + (1 - \alpha) * (\beta * OPE_p + \frac{(1 - \beta)}{Traffic_p}) \tag{2.9}$$

2.4.7 Evaluation

The evaluation method is defined as comparing the preferences of real architects with our generated results. The comparison process consists of presenting several different floor plans to architects and asking them to sort these design solutions based on circulation quality, then by comparing their results, we can find out how accurate our program is in satisfying architectural expectations. For this paper, a preliminary study with 6 architects was conducted. The participants were from Spain, the Netherlands, and Iran. Despite the fact that our proposed approach is more applicable in complex buildings such as hospitals and schools, to simplify the process of estimation for architects, home floor plans (instead of complex buildings plan) were used in this evaluation. Four floor plans, along with a relationship matrix, were presented to architects.

R	A	B	C	D
A	0	Should	Could	Must
B	Should	0	Should	Could
C	Could	Should	0	Must
D	Must	Could	Must	0

Table 2.1: A simple relationship matrix.

R'	A	B	C	D
A	0	0.5	0.25	1
B	0.5	0	0.5	0.25
C	0.25	0.5	0	1
D	1	0.25	1	0

Table 2.2: Quantitative representation of Matrix R.

Each of these floor plans is a design alternative that covers the expectations of relationship matrix to some extent. Each participant was asked to sort the input floor plans by considering the relationship matrix and other factors that he/she believes have influence on circulation. Participants were free to devote as much time as they need for sorting plans.
First, we sorted alternative floor plans through our proposed approach in which the output is a sorted list and $\alpha, \beta = 0.5$. Then we asked participants to sort floor plans and create a sorted list for presenting the order. Table 2.3 shows the results where values determine the rank of the corresponding floor plan. In order to measure the overall efficiency of our algorithm we compare the order of participants lists with our lists order. The comparison is performed through a similarity metric that measures how closes our list is to a list that generated by a participant.

$$SimDiff_{u,v} = \frac{\sum rank(u,i) - rank(v,i)_i}{MaxDifference(u,v)} \tag{2.10}$$

Where $rank(u,i)$ implies the priority of plan i in list u. In addition, $MaxDifference(u,v)$ calculates the maximum possible dis-similarity between two lists u and v.

$$MaxDifference(u,v) = \frac{\sum rank(u,i) - rank(v,i)_i}{MaxDifference(u,v)} \tag{2.11}$$

Where n is the number of plans in list u that in our case is 4, thus

$MaxDifference(u, v) = 8$.

	OA	P1	P2	P3	P4	P5	P6
Plan01	2	3	2	2	3	3	3
Plan02	3	2	3	3	4	1	2
Plan03	1	1	1	4	2	2	1
Plan04	4	4	4	1	1	4	4

Table 2.3: Evaluation results of our approach and participants.

Table 2.3 shows the $SimDiff$ for all participants. The result of our experiment is illustrated in Table 2.4. Finally, the average of $SimDiff$ scores demonstrates that our approach judge the circulation quality of a plan 62 percent similar to an architectt's mind. Also, we measured the similarity between architects using the same equation. The result of this calculation was 38.3and the higher equivalent value of our technique, suggests that our technique provides an independent method of assessing space quality that is less subject to individual bias. Each architect spent more than 30 minutes for sorting floor plans while computation time of our algorithm is only a few minutes. We believe that in multifaceted building projects our proposed algorithm not only accelerates the decision-making process, but also assists architects to prevent errors and undesirable planning results.

	P1	P2	P3	P4	P5	P6	Avg
SimDiff	0.25	0	0.25	0.75	0.5	0.25	0.33

Table 2.4: The overall difference of our proposed approach based on $SimDiff$.

2.5 Conclusion

Circulation is perhaps the most significant component in defining and expressing spatial form and function. Through a circulation path, a semantic relationship between spatial units is created which not only defines the quality of accessibility, but also influences other spatial quality metrics such as privacy. In this paper, we attempt to measure the circulation quality in interior spaces. The study is founded on asking ourselves how an architect can select the best solution among different creative design alternatives in terms of circulation functionality. Our proposed metric does not take into consideration changes in floor level when measuring the weight between space units. As a further line of research, it would be extremely interesting to measure the influence offloor height on path weight for those buildings

containing stairs and ramps. Another promising direction is measuring the quality of circulation based in some particular situations such as hospitals and schools. In addition, we can develop this domain for analysing the quality of space according to other metrics such as privacy and illumination.

Chapter 3

MEASURING THE PRIVACY AND VISIBILITY

3.1 Introduction

Privacy has been discussed in different contexts such as law, philosophy, society, biology and architecture [Georgiou, 2006]. Privacy determines how and to what extent information about an individual or a group should be communicated to others [Westin, 1968]. In sociology, the built environment is seen as the extension of the human epidermis, and therefore, each broadly defined space is interpreted as an extended membrane of the body with the capability of communicating to the environment [Acquisti et al., 2015]. Using a similar analogy, physical boundaries can be conceived as filters that control the type and amount of information which is received by human senses.

Traditional models of architecture classify spaces in a building into two types based on the privacy factor [Riley and Morris, 1999], namely, public and private. However, [Georgiou, 2006] suggested different degrees of privacy for spaces base on their geometrical attributes. Although this approach supports a wider range of privacy for spaces, still all the points of a space have the same privacy. The state of architectural elements, such as doors and windows and the arrangement of furniture do not contribute to privacy. In this chapter, we present a finer granularity of privacy by measuring the privacy degree in each point (each possible coordinate) of the building instead of each space, resulting in a more accurate measure of privacy for a layout or a building. Moreover, we consider privacy as a dynamic property of the space since the privacy degree of a point in the space depends on the state of architectural elements in the scene. For instance, a window, which is a dynamic architectural element, can change the privacy level by changing its state from *closed* to *open*.

Privacy can be interpreted as the regulation of social contact with incoming information and filtering out of unwanted crowding within a space [Demirbas and Demirkan, 2000]. All human senses, through which the surrounding environment is perceived, can influence privacy. Among the five main senses, sight and hearing are those mostly considered in architecture as regulating the visual privacy and vocal privacy, respectively. Visual privacy is defined as the extent to which someone can restrict his/her visibility from other people (viewers). A 3D space enjoys vocal privacy when someone is not distracted by another people's voice inside the space.

In this thesis we only consider visual privacy while vocal privacy is an interesting topic of future work.
Visual privacy can be measured through a 3D spatial volume around the viewer, called *isovist*, in which the viewer is visible from each 3D point in the volume. Thus, the largest part of visual privacy computation overlaps with visibility calculations, that have been the focus of a lot of research in computer graphics. However, visibility algorithms in computer graphics are applied in shadow computations and rendering optimization, while the main goal of calculating visibility in architecture is measuring the effect on human perception of visible elements in the scene.
In the remaining of the chapter we first discuss related work (section 3.2), before describing our proposed approach (section 3.3). This approach was applied in the Interactive Layouts Recommender System discussed in chapters 5 and 6, where the overall results are evaluated. The final section of the chapter (section 3.4) deals with conclusions.

3.2 Background and Related Work

In this section we review studies related to privacy and visibility calculations. The different perspectives adopted by different fields originate a wide range of solutions to visibility problems [Durand, 2000]. In this review we divide the works into two categories based on their applications: computer graphics and architecture. The former discusses visibility methods in computer graphics while the latter elaborates on the notions of privacy, visibility and visual openness in architecture.

3.2.1 Visibility in Rendering

Visibility computation is the process of deciding which surface can be seen from a particular point of view. It is a crucial problem in computer graphics applications [Bittner and Wonka, 2003]. The visibility problem has recently been receiving more attention by researchers in computer graphics, as the ever-increasing size of 3D data sets challenge the efficiency of traditional algorithms. In the following we present an overview of different visibility methods that have been applied to render virtual scenes.

Shadow Mapping

In [Woo et al., 1990] a shadow is defined as: "... *a region of relative darkness within an illuminated region caused by an object totally or partially occluding the light.*"

Shadows in synthesized images of virtual scenes convey the reality and clarify the spatial relationships among scene elements [Amanatides, 1987]. In fact, shadows provide significant clues of relative depth between objects. Computer graphics tools use a particular type of map called shadow map which provides the visibility attribute of polygons, from the light source point of view into grayscale colors. Visibility algorithms deal with both hard and soft shadows. The efficiency of visibility algorithms in real time rendering is critical when the light is moving since the visibility testing has to be performed in every frame.
Hard shadows and *soft shadows* are two types of shadows that appear in the literature. Calculation of hard shadows essentially means decicing whether a point is in the invisible area from the light source point of view or not. Hard shadows are usually generated by a point light source. Soft shadows are produced by light sources with finite extent, as points of surfaces can receive a fraction of the light. Later, we use the concept of soft shadows to measure the privacy of the points which are seen partially by a viewer.
One the first literature surveys of shadow computation methods [Woo et al., 1990] discussed three types of factors to be considered in the choice of the shadow mapping method: the rendering method applied, the type of modeling primitives and the required level of accuracy. A more recent survey by Liu et al. [Liu and Pang, 2009] discusses the differences between two important shadow mapping techniques: projection and volume algorithms. A rich comparative analysis of real time hard shadow mapping techniques is presented in [Scherzer et al., 2011], while a similar analysis about soft shadow mapping methods has been presented in [Hasenfratz et al., 2003].

Hidden surfaces removal

Visibility optimization was the main focus of early real time rendering research: removing the hidden surfaces from the render process was intended. Hidden-part removal algorithms have been also applied in non-photorealistic rendering (technical illustrations and cartoons). Hidden surfaces removal has been also addressed in object precision methods. For instance, it was proved in [McKenna, 1987] that the complexity of a view can be $O(n)$ where n is the number of edges in the scene.Two categories of algorithms were distinguished, *Image precision* and *object precision*. The former algorithms focus on the rasterized image by sampling the visibility at pixel level. The latter are applied to generate a *visibility map* based on the arrangement of 3D objects in the scene. The early survey by [Sutherland et al., 1974] provided a comprehensive categorization of several techniques about hidden surfaces removal. Wexler et al. presented more recently a GPU based surfaces removal algorithm that outperforms traditional CPU based algorithms [Wexler et al.,

2005].

z-buffer

Later, by emergence of rasterize devices z-buffer based algorithms were proposed with the intention of faster detection of visible surface. The z-buffer first introduced by [Gouraud, 1971] as one of the most efficient and simplest visibility algorithm that requires a particular hardware implementation. The algorithm assigns a depth value to each pixel of the image. Therefore, objects that are collided by other objects can be easily identified by a depth test pass from the camera point of view. Although, applying z-buffer algorithms seems appropriate for visible surface determination, ray casting methods are more efficient to evaluate the visibility along a single ray. In our proposed method we apply a ray shooting method to measure the visibility level of point in a space from a particular vantage point.

Global Illumination

Global illumination is defined as the rendering obtained by means of a physically based simulation of the light that is scattered in a 3D virtual scene. Calculating intersections between rays and surfaces is a major part of global illumination calculations. Each surface is illuminated by the light sources (direct illumination) or by the reflected light coming from other surfaces in the scene (refractions could play a role as well). The visibility algorithms are used to specify which surface is being viewed from which surfaces or light sources. In [Dachsbacher et al., 2007] an *implicit* visibility method is applied to provide fast interactive indirect lighting through directional discretization. The presented rendering equation takes into account both radiance and anti-radiance, while enabling the treatment of visibility implicitly. The main limitation of this approach is the large usage of memory as a result of the directional data structure. Engelhardt [Engelhardt, 2013] has recently discussed the efficiency of recursive visibility evaluation methods in *ray tracing*-based rendering algorithms. A k-means clustering algorithm called *isodata* was presented to group polygons based on a distance function. The main contribution of this paper is the proposal of a fast and simple technique for identifying potentially visible clusters.

Visibility in virtual environments; view point selection

Visibility can also be applied to regulate and analyze the quality of either visible objects or the viewer in virtual environments; one example is view point selection. Although in visual effects and production animations the camera is mostly placed

by digital artists, in some video games and real time 3D applications the camera needs to be positioned automatically based on the application. For instance, in first person shooter games, the camera illustrates the entire scene from the user's point of view except the part that is occluded by the player body. Therefore, if a certain object is supposed to be viewed by the user at a certain time slot, then the object should not be located at a position where it is occluded by the player body.
Koenderink has developed a structure called *aspect graph* to measure the visual potential of objects in the scene. Each node in the aspect graph represents a view point from a particular position, while each edge in the graph a visual transition between two view points. A wide range of works that are inspired by the aspect graph concept have been reviewed in [Eggert et al., 1992, Schiffenbauer, 2001]. Moreover, a rich psychological analysis of the camera point of view on player perception is presented in [Yannakakis et al., 2010].

3.2.2 Architecture

In architecture, visibility is usually discussed as a measure of spatial quality. For instance, a case study through analysing the architectural qualities of the a design studio in [Demirbas and Demirkan, 2000] is presented that follows three purposes: identifying privacy regulation, evaluating different types of privacy introduced by [Pedersen, 1979] and providing a comparison analysis about privacy preferences of different sexes. In a similar study, the level of privacy that is perceived by people in a office is analyzed based on the size and type of screen [Little et al., 2005]. The results show the smaller screens with partitions are rated significantly higher than screens with larger size without partitions.
A quantitative metric of open space observation based on the points of view of building's windows is presented in [Fisher-Gewirtzman and Wagner, 2003] . The proposed metric measures the visibility and permeability of spatial configurations in urban design that can be applied to rank different layout arrangements based on visual openness.
In [Georgiou, 2006] six studies are proposed to produce a set of graphs (assembly graph, visibility graph and proximity graph) with the intention of evaluating hypothesis about privacy and configuring a architectural program. The program later is applied to measure the privacy of rooms in a building layout. The main limitation of the presented program is setting accurate weights of privacy factors which can be varied based on the contextual aspects of the planning project.
The effect of visibility on the movement pattern and visual understanding of the visitors inside a museum area is discussed in [Wineman and Peponis, 2010]. For instance [Wineman and Peponis, 2010] measured whether the visibility of other visitors affects what visitors tend to watch. Other type of studies, in a more clas-

sical vein, investigate how many cameras are need to guard a museum [O'rourke, 1987, Borrmann et al., 2013, Bärtschi and Suri, 2014]. This problem, known as the art gallery problem, has been shown to be in the category of NP hard problems [Durand, 2000].
As before, visibility methods are also applied in architecture to measure the visual privacy. In architectural terms, privacy is defined as: "*the property indicating the amount of information which is communicated through the boundaries to the surroundings*". Next, a topological based model is proposed to analyze and synthesize privacy in early stages of the architectural process.
Mustafa et al. [Mustafa et al., 2010] have defined privacy as a dynamic property of the space that can be regulated based on the time and the type people (who) are in the space. They have presented a morphological analysis of a sample of both traditional and modern house in Erbil City. It has been concluded that the traditional sample offers a higher level of privacy since accessing through the spaces is carried out by particular control spaces.
Indraprastha et al. [Indraprastha and Shinozaki, 2012b] modeled visual openness of windows in a building through analyzing the visible area from each window. Three factors are considered in measuring the privacy: distance, transparency ratio and view angle. We consider similar factors to measure the privacy, however, instead of selecting a set of particular points, all coordinate points of the space (as a result of discretization method) are taken into account in privacy calculation. Thus the distance and angle of view are measured between two points of the discretized space, instead of a point and opening, resulting in a more accurate and richer analysis of the privacy.

3.3 Proposed Approach

3.3.1 Visual Perception

The main task of visual perception is the continued capturing of visual information coming from different points surrounding us [Bittermann et al., 2006]. Spatial experiences and cognition in an architectural space are mainly based on the information gathered through the visual perception system. Having a rich visual model is relevant when measuring privacy, since it can inform on how the environment is being seen from the human vantage point. Furthermore, not all the objects visible to the eyes, are considered as *seen* objects in the brain. Some objects might be overlooked by us humans, based on the configuration of the space and the state of the mind when the visual information is received.
According to [Ciftcioglu et al., 2006, Bittermann and Ciftcioglu, 2008] we are able to remember only objects that we *saw*, not everything that was visible to us.

This throws uncertainty characteristics that differentiate the human vision system from an optical device like a camera. Due to the brain complexity, it is difficult to measure or model how humans memorize a scene; however, the geometrical configuration of the space influences the perception. It can be concluded that not-seen objects, those that are not perceived by a human at a certain moment, less likely violate what someone perceives as part of his/her private boundary. Therefore, the arrangement of space units in a building not only shapes the volume of visible objects surrounding us, but also defines to which level we feel and perceive privacy.

3.3.2 Measuring Visibility from a Viewer Vantage Point

In this section, a method is proposed to measure the overall visibility at a given view point based on the visibility of the point from other points in the space. Thus, the more visible the point is, the less private is that point. However, as mentioned above, not all the visible objects contribute in the same way to the understanding of the human of the space. People are more likely aware of objects located nearby than of those which are placed far away. Moreover, the awareness level can vary based on the angle of view. According to [Ciftcioglu et al., 2006] the visual awareness about the objects which are located in the center of the view field is larger than about those in peripheral regions. It goes without saying that we have much less awareness about the objects are placed behind our head. Hence, we add contributions related to distance and angle in the visibility calculation. In Figure 3.1 we illustrate this phenomenon through colorizing space unit points. There, the layout basement is divided into non-overlapping cells. The cells which are visible from a viewer point of view are shown with a color based on their distance and angle from the viewer. The greener a cell is, the more likely is to be considered as a *"seen"* cell by the viewer. In the equation below the visibility from the view point of a viewer i is calculated.

$$V_i = \sum_{j=0}^{n} v_{i,j}/n$$

$$v_{i,j} = f_{i,j} \times \Delta_{i,j} \times \Theta(lookAt_i, j)/180.$$

$$f_{i,j} = \begin{cases} 1 & j \text{ is visible in viewer } i\text{'s frustum} \\ t & j \text{ is visible in viewer } i\text{'s frustum through a transparent element} \\ 0 & \text{otherwise} \end{cases}$$

$$0 < t < 1$$

Figure 3.1: The different level of environment perception of a viewer depending on distance and angle of view.

where V_i is the overall visibility from viewer i, $v_{i,j}$ is the visibility of point j from i, t is the *transparency ratio*, n is the number of points in the frustum of viewer i, $\Delta_{i,j}$ is the distance between i and j, and f determines whether the point j is visible in the frustum of viewer i. It is worth nothing that transparency ratio define transparency level of an object, thus the more transparent is the object the bigger t is considered for the object. Figure 3.2 visualizes the visible cells through the transparent window with a blue-violet color template.

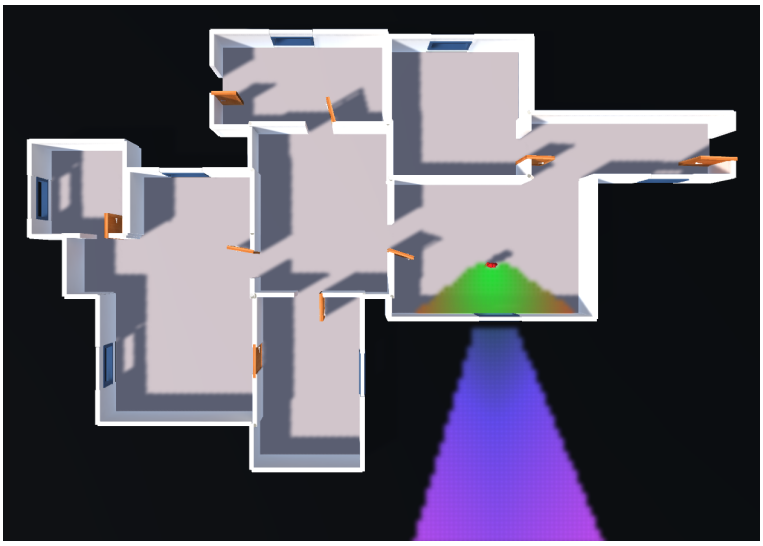


Figure 3.2: Visualizing the visible points through transparent window by blue-violet template color

3.3.3 Measuring the Overall Visual Privacy of a Space Unit

As mentioned in Sec 3.3.1 we apply a tessellation based approach to convert the continuous physical space of the floor layouts into a discrete space. The tessellation technique creates a grid of cells that have the same shape and size. The physical space of a layout is tessellated into a finite number of non-overlapping cells and the privacy value is measured at the center of each cell, separately.
In architecture, space units can be distinguished and categorized based on their privacy level [Georgiou, 2006], which is an architectural quality. The privacy level of a cell not only depends on the number of visible cells from that cell view point, but on the type of the visible points as well. For instance, it may be preferred to be seen by people in the same office, but rather less by staff in a neighboring office. Therefore, for a given cell q in a space unit $\mathcal{P}^q$ within a layout or a floor plan $\mathcal{L}_a$, we define three types of cell visibilities:

- v_q^o visibility from a point outside the floor plan. We call that space O.
- v_q^s visibility from a point inside $\mathcal{P}^q$.
- v_q^d visibility from a point in another space unit inside the floor plan.

Therefore, the total visibility of a point q is determined as:

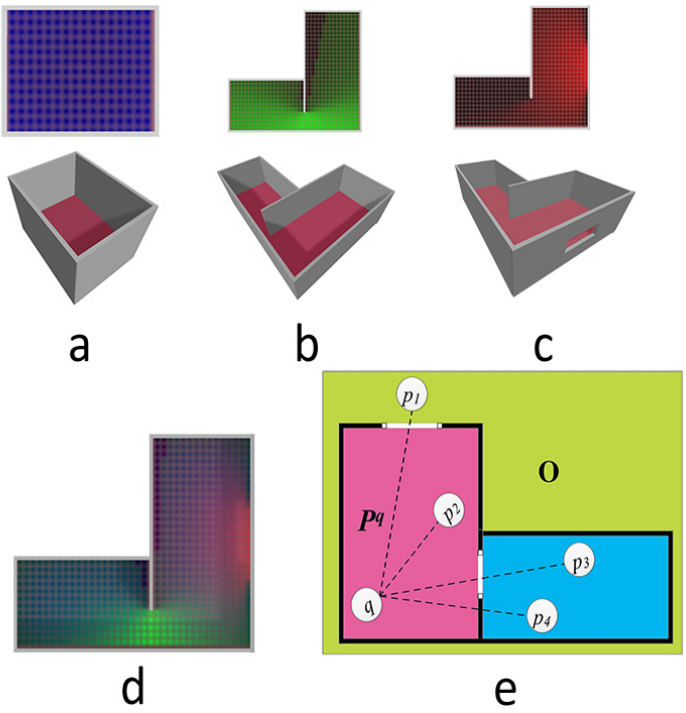


Figure 3.3: (a), (b) and (c) different levels of privacy are shown, while (d) represents overall privacy for the layout. In (e) q in $\mathcal{P}^q$ is viewed from p_1 outside the layout, p_2 inside it, and p_3 inside the layout but not in the same unit; q and p_4 cannot see each other because of a wall between them.

$$v_q = (w_o v_q^o + w_n v_q^d + w_s v_q^s)/n_G$$

where

$$v_q^o = (\Sigma_{i \in O}\ f_{i,q} \times \Delta_{i,q}/\Delta_{max})/n_o$$

$$v_q^s = (\Sigma_{i \in \mathcal{P}^q}\ f_{i,q} \times \Delta_{i,q}/\Delta_{max})/n_s$$

$$v_q^d = (\Sigma_{i \in \mathcal{L}_a \setminus \mathcal{P}^q}\ f_{i,q} \times \Delta_{i,q}/f_{max})/n_d$$

$$f_{i,j} = \begin{cases} 1 & j \text{ is visible in viewer } i\text{'s frustum} \\ t & j \text{ is visible in viewer } i\text{'s frustum through a transparent element} \\ 0 & \text{otherwise} \end{cases}$$

$$0 < t < 1$$

$$n_G = n_o + n_s + n_d,\ w_o + w_n + w_s = 1$$

and n_G is the total number of cells, n_o the number of cells outside the floor plan, n_s the number of cells inside $\mathcal{P}^q$, n_d the number of cells inside all space units of the floor plan except for $\mathcal{P}^q$, $\Delta_{i,q}$ is the distance between cell i and cell q, while Δ_{max} is the maximal distance between two arbitrary cells in the site; v_q ranges between 0 and 1.
In Figure 3.3 we illustrate different types of visibilities of cells by which v_q^o is mapped to red, v_q^s to blue and v_q^d to green. Figure 3.3.a visualizes v_q^s for cells inside the space unit. Figure 3.3.b shows the visualization of v_q^d of each cell. In Figure 3.3.c, a window is added which allows some cells to be viewed from the outside. Thus, v_q^o is visualized for each cells that can be viewed from the outside. Figure 3.3.d illustrates the combination of all three types of visibility.
The privacy quality of a point q is determined as:

$$\mathcal{P}r_q = 1 - v_q$$

Finally, the total privacy quality of the layout $\mathcal{L}_a$ is defined as the sum of the privacy quality of all points in $\mathcal{L}_a$:

$$\mathcal{P}r(\mathcal{L}_a) = \Sigma_{q \in \mathcal{L}_a} \mathcal{P}r_q$$

3.4 Conclusion

In this chapter we have discussed the significance of visual perception and privacy within the architectural design process; in particular, privacy is a sort of complementary of visibility. Taking into account this discussion, we have proposed two computational metrics, one for the visibility from a viewer vantage point and another one for the overall privacy of a space in a 3D layout. The visibility is calculated based on the distance, and angle of those points which are inside the viewer point of view. The overall privacy of the space is calculated through adding the privacy of the central points in a tessellation of the space. The visual privacy of a point in the space not only takes into account the visibility of the point from the other points in the layout but also the space unit the point belongs to. The measures we propose take into account the architectural perspective, but are still early attempts. It is likely that we will be considering to have different weights for different space units according to their privacy priority.

Chapter 4

GENERATING RECTILINEAR LAYOUTS BASED ON SUBDIVISION METHODS

Title	Recommendation of Floor Plan Layouts Based on Binary Trees
Authors	Arash Bahrehmand, Alun Evans, and Josep Blat
Conference	EG-ICE: TOWARDS NEW GENERATION SMART BUILT ENVIRONMENT, 2014
Keywords	Floor Plan, Binary Tree, Non-Rectilinear Space, Circulation.
Year	2014

Recommendation of Floor Plan Layouts Based on Binary Trees

4.1 Abstract

Interior space planning, as a creative part of the architectural design process, requires several conflicting criteria to be taken into consideration to find optimal solutions. This paper presents an automatic approach for suggesting floor plans to architects based on some dimensional and topological input constrains. We apply two different top-down and bottom-up methods of tree construction generate all possible arrangements of rectangular and non-rectangular spaces. Dimensional and Topological requirements of the floor plans are fulfilled through the architects input constraints. Efficiency of the proposed recommendation method is investigated by evaluating some real world scenarios, the results of which confirm that it can indeed help architects to consider more alternatives in approaching the final optimal solution.

4.2 Introduction

Interior space planning is a challenging part of the architectural design process, as it requires several conflicting criteria to be taken into consideration to find optimal solutions. In order to design a floor plan that satisfies various pre-defined requirements of the project (or client) an architect must deal with several factors involving layout and the relationship between spaces. Space layout planning must satisfy both artistic and logical requirements. Meeting these needs is even more difficult since there are neither accurate initial constraints, nor clearly formulated final goals [Homayouni, 2000].Therefore, the time-consuming process of trial and error to reach the best (final) set of solutions is undeniable. In this sense, there is a need for novel space planning methods, which use computational techniques to efficiently find the best alternative. Computerized space planning methods attempt to find a computational method in order to approach the best solution among a large space of possible solutions rapidly. The efficacy of a layout plan depends on the functionality of each space unit on one hand, and the functional relationships between them on the other. In this sense, having a clear perception of the characteristics of people who use the resulting building affects the performance of layout design. According to [Homayouni, 2000], computational constrains can be divided into two categories, dimensional constrains and topological constrains; the former is focused on the geometric attributes of the space, and the latter determines the

types of relationships between spaces. Although several studies for automatic generation of city plans have been presented[Parish and Müller, 2001, Greuter et al., 2003], intelligent generation of floor plan has not received the same level of attention, perhaps due to the difficulties (ambiguities) in defining the final solution. The objective of the research presented in this paper is to assist architects in generating all possible alternative solutions which can be produced according to some input constrains, and sorting/ranking them according to given metrics. Our method, like real world architectural practice, uses the dimensions of architectural elements and relationships between them as inputs. Two construction top-down and bottom-up methods of tree constriction are applied to arrange space units inside the base area. The top-down approach divides the base input shape, in our case it is a rectangle, into sub-rectangles, in a recursive manner, to generate all possible arrangements of rectangular and non-rectangular spaces. The bottom-up approach then creates space units separately and then generates floor plan by assessing the possible combination of the space units. Both the 2 splitting and combination methods use the central concept of binary trees (Fuchs et al. 1980) to generate alternatives. For the last step, generated plans are sorted based on the circulation quality metric, as presented by [Bahrehmand et al., 2014a]. Novel contributions of this paper can be summarized in using a modified version of binary tree to generate all possible solutions, non-rectangularity in interior and exterior spaces, and providing flexibility for the architect to specify constraints. The reminder of this paper is organized as follow: in section 2 recent studies generating floor plans are reviewed. In Section 3, a detailed description of the proposed approach is provided in eight subsections. In Section 4, we evaluate the efficiency of our approach through experimental testing. Finally, conclusions and future work are discussed in Section 5.

4.3 Related Work

Space planning techniques often use spatial allocation programs to organize architectural elements with the intention of finding the best way to satisfy the constrains of a problem. [Regateiro et al., 2012] present an approach based on topological algebras and constraints satisfaction techniques that helps an architect in finding solutions for a layout design problem. Different computational methods are presented for assisting architects during the early stages of conceptual design, each concentrating on different aspects of space planning [Indraprastha and Shinozaki, 2012a, Wiener and Franz, 2005, Readinger, 2002]. However, most of these methods remain in academic stage and have not had notable achievement in the marketing stage [Leblanc et al., 2011, Merrell et al., 2010, Homayouni, 2007].In [Marson and Musse, 2010] a real time technique for providing floor plans is presented.

Floor plans are generated based on a squarified treemaps algorithm which is introduced as an efficient method for organizing hierarchical structures. The presented method uses several input constraints such as width and height of rooms. Although this approach generates several floor plans, it suffers from two main drawbacks. First, the lack of references about statistical information used to satisfy the topological constraints and properties, which need to take into account client requirements and site specifications (thus, making decisions only based on analyses of a few floor plans is not appropriate). Second, the presented tool does not support non-rectangular plans. Merrell et al. present an end-to-end approach to generate residential floor layout [Merrell et al., 2010]. The input of the presented tool is a list of high-level client requirements such as the number of bedrooms and bathrooms, and approximate square footage. Finally, generated plans are optimized based on some space quality metrics. The input information is expanded into a Bayesian network trained on some analyzed data from [Hanley Wood, 2007]. The main limitation of this approach is that the final shapes of the whole boundary of generated plans are not predictable, therefore it is quite possible to have a plan which does not satisfy site specific constrains. In addition, the method of arranging spaces is not clear. In [Knecht and Koenig, 2010], a k-D tree based approach is presented which analyses the space layout problem in three levels of urban neighborhood, building structure and floor plan. One of the advantage of their system was adjusting and modifying the floor plan in real time. However, the whole boundary of plans is kept in a rectangular form. Furthermore, the concept of floor planning is argued in VLSI to provide early feedback about design complexity of circuits [Wong and Liu, 1989]. Two widely approaches of planning in VLSI are simulated annealing and analytical formulation using B*tree for slicing and combing modules [Chen and Chang, 2006].

4.4 Proposed Approach

Space planning is a special form of architectural design concerned with defining a function driven structure for an empty space. Our approach tries to delineate the functionality of different parts of a given empty space by placing walls to define sub-spaces, and connecting them via connector elements.

4.4.1 Inputs

In order to manage and resolve various design criteria and project requirements, an architect is asked to convey his/her general thoughts about building functionalities in the form of system inputs. In our intelligent space planer, inputs are applied

as predefined constraints falling into two categories: Dimensional constraints and Topological constraints. Dimensional constraints concern the geometrical definition of a space unit. In this step, our method accepts the width and height of the base empty space and sub space units. Despite the fact that higher quality solutions are generated with more accurately applied input constraints, it is not always possible to determine the precise dimension values by architects since in the early stages of the design he only has an abstract perception of the final result Therefore, we present another parameter called *flexibility* level defining the current architects confidence with respect to a certain dimension value. Table 4.1 shows a sample of input constraints. Topological constraints define the spatial relationships between space units. In order to determine the topological circumstances of the floor plan, we define two main: *Adjacency* and *Accessibility*. The former only specifies which space units should be considered as direct neighbors while the latter defines the accessibility type of space pairs:

Adjacency. The adjacency constraint is defined through two parameters: Space unit pair and Location. The first parameter provides the architect with the capability of defining two spaces as direct neighbours and the second one specifies the location of a space unit in the final floor layout. An Adjacency constraint is applied when two spaces share a wall without necessarily being connected by door. The location parameter receives one of these four values as input: *north*, *south*, *east* and *west*. In addition, the location parameter can be set as *null*.

Accessibility. Two spaces are connected to each other by a connector element. In our system, two types of accessibilities are defined: *interior accessibility* and *exterior accessibility*. The connector element of interior type has two parameters, *type* and *width*, while type can be *door* or *open wall*. By connecting two spaces by an open wall we have the possibility of generating non-rectangular shapes such as L-shape and U-shape spaces. On the other hand, the exterior type defines the entrance of the building. In fact, entrance is a connector element that has two values: *location* and *width*. Figure 4.1 illustrates the difference between door and open wall.

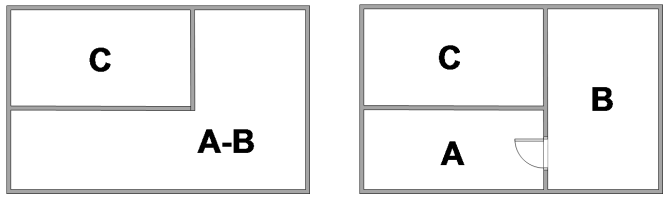


Figure 4.1: Connecting A and B by door and open wall

4.4.2 Constructing Binary Tree

Given the set of inputs, we need to construct a data structure, in order to maximize the number of possible layout solutions. In this work, we use the concept of a Binary Space Partitioning (BSP) tree for recursively subdividing an n dimensional space into convex subspaces, resulting in a tree data structure. BSP, as an inherited concept from computational geometry, has been widely used in computer graphics due to its efficiency and large coverage of solutions [Fuchs et al., 1980]. We apply this concept to divide and combine spaces based on dimensional guidelines.

Space Unit	Width	Length	Flexibility	Interior accessibility	Exterior accessibility	Adjacency
Base Area	200	135	-	Exterior-Door	True - North	null
Bed Room1	75	70	0%	Foyer	False	null
Bed Room2	70	70	0%	Foyer	False	null
Living Room	70	100	0%	Foyer	False	null
				Dining	False	null
Bathroom	75	35	0%	Foyer	False	null
Kitchen	55	85	0%	Dining	False	null
Foyer	25	60	0%	Bed Room1	False	null
				Bed Room2		
				Living Room		
				Bathroom		

Table 4.1: Constraints that should be considered for a sample layout

In our binary tree, each non-leaf node corresponds to a wall that divides the given space into two new subspaces and each leaf node is a space unit from input space units. Each non-leaf node has two main attributes: ratio and orientation. Ratio concerns the position of the division and its value ranges between 0 to 1. Orientation determines whether the division is horizontal or vertical. For example, if ratio = 0.3 and orientation = vertical then the given space is divided by a vertical wall while the sub space in the left side of the wall has area of 30concept is inherited from VLSI floor planning [Sait and Youssef, 1999]. Figure 4.2 illustrates the result of applying a sample binary tree on a given rectangle. Generating all possible binary trees, according to dimensional constraints, helps us to increase the state space of floor plan alternatives. In order to generate all possible alternative solutions three variable factors are applied: generating all possible structurally different BSP trees, generating all permutations of space units for leaf nodes, generating all feasible combinations of orientations for non-leaf nodes. Our tree construction method is described in algorithm below.

```
(1)   Foreach (BTree in UniqueBTrees)
(2)       Foreach (SpaceUnitSequence in SpaceUnitSeqs){
(3)           Reconstructing BTree through SpaceUnitSequence ;
(4)           CalculatingRatio (BTree ) ;
(5)           Foreach (OrientationSequence in OrientationSeqs) {
(6)               Reconstructing BTree through OrientationSequence;
(7)               If (BTree is valid)
(8)               FinalBTrees ← BTree
(9)           }
(10)      }
```

In order to initialize the algorithm, first, all structurally different binary tree templates with *n* leaves are generated such that *n* is number of space units [Proskurowski, 1980]. We store all the resulting templates in a list of binary trees, named *UniqueBTrees*. The list of space units is assumed as a sequence of *n* (string) names such that each name corresponds to a leaf name of the binary tree. Therefore, in order to cover all possible order leaf nodes for a tree, we calculate all the permutation of name sequence and store it in a list, named *SpaceUnitSeqs*. A binary tree with *n* leaf nodes has *n-1* non-leaf nodes. As is pointed out before, a non-leaf node specifies the wall direction, which is either *horizontal* or *vertical*. Finally, instead of generating all the possible combinations of orientation sequences for non-leaf nodes, a bottom-up method is applied, in section 4.4.4, providing only feasible orientations for a given binary tree.

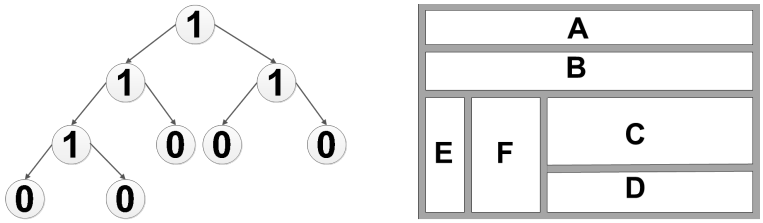


Figure 4.2: A binary tree representation of a floor plan

4.4.3 Top-Down Approach

The top-down approach starts from the highest-level node of the binary tree and proceeds through to lower levels. In this technique first all possible binary trees are generated then the splitting process start from the root node and breaks down the base rectangle to smaller space units. In order to generate all possible alternatives

a semi-brute force approach using two forloops (see Section 4.4.2) are applied to loop through all unique binary trees and sub-spaces name sequences. In line (2), a space unit sequence, *SpaceUnitSequence*, is bound to the binary tree template, and then the ratio and rotation of non-leaf nodes are calculated in lines (4) and (5) respectively. Finally, the tree is added to the *FinalBTrees* if it satisfies dimensional and topological requirements through line (7). Finally, FinalBTrees stores the list of final binary trees for creating the floor plans. In each iteration a new binary tree is generated, line (8), and is added to the FinalBTrees.

4.4.4 Bottom-Up Approach

In order to generate the ratio and direction values for non-leaf nodes bottom-up approaches are applied. The ratio is calculated by dividing the area of the left child by the right one. A recursive function, called *CalculatingRatio*, is applied to calculate the ratio in a bottom up manner.
The function starts to calculate the ratio for non-leaf nodes at level *D-1*, through leaf nodes area (*D* is the Depth of the tree). Then it climbs the tree recursively to calculate the ratio for the remainder of non-leaf nodes. Figure 4.3, illustrates how ratio is calculated for a tree where the values below the space units names are space units dimensions, on the left side, and space units areas, on the right side.

4.4.5 Flexibility

. Flexibility creates a range of allowed variations for each space unit, but also causes infinite alternatives for the ratio of a non-leaf node. In Figure 4.4, dot lines are the maximum and the minimum boundary of A and B, and there are several possibilities for combining them. To simplify explanation, here we only assess the possibility of three alternatives for two given spaces. These alternatives are as follows: $A_{mid}B_{mid}, A_{min}B_{max}, A_{max}B_{min}$. In the last loop, the orientation of the each non-leaf is specified and the final BTree will add to the FinalBTrees if it satisfies dimensional constraints.
In order to verify a BTree, the base rectangle, root node of BTree, is broken down into subspaces based on ratio and orientation of non-leaf nodes. Then a BTree is marked as *valid* if it can generate appropriate leaf nodes after cutting the given space such that leafs dimensions are matched with inputs. For each leaf node, we testify whether the width and the height are included in the dimensional range of that space unit in input constraints. Therefore, because of flexibility parameter, having both the vertical and the horizontal cuts for a non-leaf node is possible. In order to calculate the direction values for non-leaf nodes a *feasibility checking* method is applied recognizing feasible orientations for a given binary tree. Hence

the output assign all non-leaf nodes one of the following values: *V* , *H* , *VH* , *False* . The bottom-up approach starts from leaf nodes and climbs the tree until determining the possible orientation of the root.

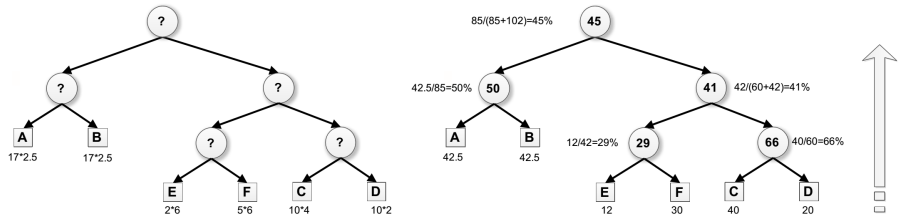


Figure 4.3: The right tree is the result of applying CalculatingRatio on the left tree

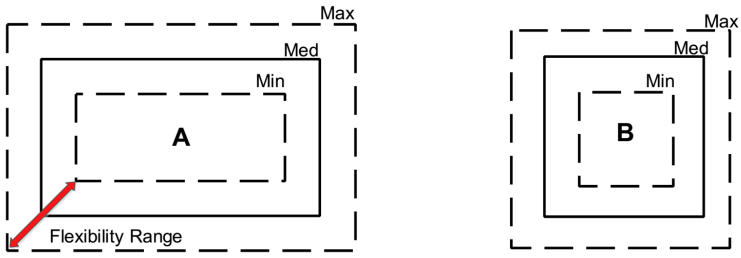


Figure 4.4: The only possible combinations are $A_{mid}B_{mid}$

4.4.6 Arranging Based Neighbours Constraints

The position of a connector element is determined by the non-leaf node which shares a wall between two given space units. The shared edge is the common parent of two space units. Since the common parent may be shared between more than one space unit, e.g. the root node in Figure4.5, we need to figure out which part of the wall is intersected by the given neighbor spaces. Intersection area can be simply calculated because coordination values of child nodes are stored on the non-leaf nodes. Then, based on the type of connector element, either a door is placed or the shared wall is removed.

4.4.7 Non-Rectangularity

As well as interior non-rectangularity via open walls, exterior non-rectangularity can be generated by simply not drawing certain space units in the base rectangle.

We define a new type of space unit, named *null*, in order to specify which spaces should not be drawn in drawing process. Figure 4.6, (b), illustrates how a non-rectangular floor plan can be drawn by removing null space units. null spaces can be defined by architect as input data or automatically calculated by the system. If the user does not specify null spaces, and the total area of space units is less than the base input rectangle, then the system automatically divides the remaining area to some null space units.

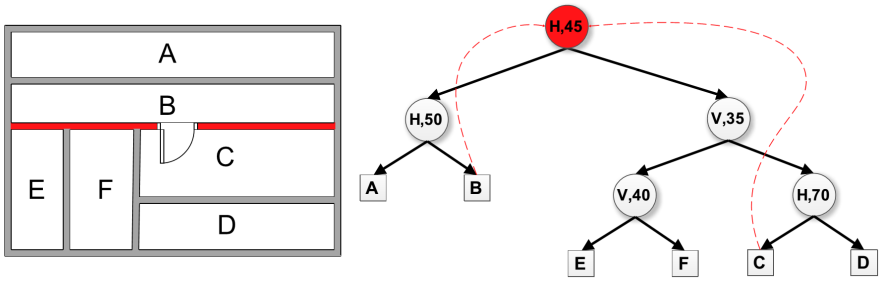


Figure 4.5: The only possible combinations are $A_{mid}B_{mid}$

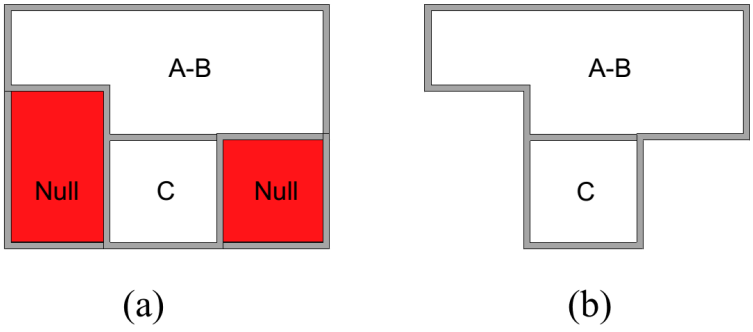


Figure 4.6: The only possible combinations are $A_{mid}B_{mid}$

4.4.8 Removing Identical (Repeated) Plans

A generated floor layout may correspond to more than one BTree, as a result of differences in the order of cutting [Wong and Liu, 1986]. This usually happens when a node and one of its children has the same cut orientation. In order to prune redundant trees from solution space we apply a filtering method just before locating doors and drawing 3D walls.

4.4.9 Sorting based on circulation quality

Due to the large number of floor plans, in some cases, the process of finding the best one becomes a difficult task, and therefore, we require a method to sort floor plans based on some quality metrics. In this paper, floor plans are sorted based on a circulation metric as presented by [Bahrehmand et al., 2014a]. For the calculation of the metric the relationship matrix and dimensional attributes of the floor plan are needed. The former can be easily extracted from the input table, while the latter is calculated separately for each floor plan.

4.5 Experiments and Evaluation

In order to implement our algorithm we developed a software application using Unity3D(Unity 2014). The application accepts a text file (in a table format) as input and draws floor plans within a 3D scene. Floor plans are visualized via 3D walls, such that an architect can still modify the walls height, width, and position after drawing process.

4.5.1 Experiment 1

In order to show how different parameters can assist architect to achieve various alternatives, we generate floor plans, for a base rectangle of 200*135, based on 4 different scenarios.

Scenario 1. In this scenario, we use a modified version of Table 1 as the input, in a way that all the topological constraints are ignored. The number of generated floor plans is 376, samples are shown in Figure 4.7.

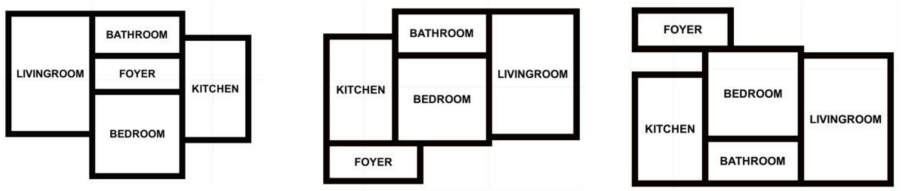


Figure 4.7: Samples of Scenario 1

Scenario 2. In this scenario, interior accessibility constraints are added to the previous experiment resulting in a large reduction in the number of feasible alternatives. The number of generated floor plans is 44 and samples are shown in Figure 4.8.

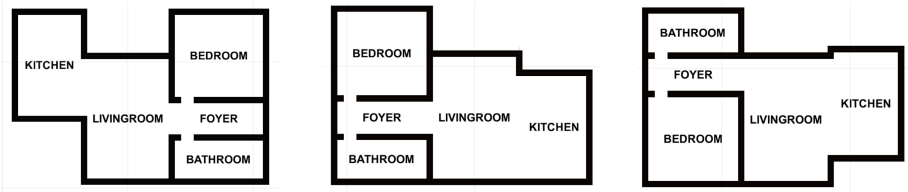


Figure 4.8: Samples of Scenario 2

Scenario 3. In this scenario, the flexibility values of living room and kitchen are increased. The number of generated floor plans is 102 and samples are shown in Figure 4.9.

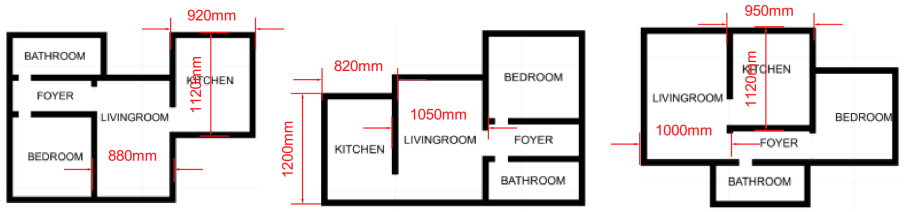


Figure 4.9: Samples of Scenario 3

Scenario 4. In this scenario, we use all the constraints from Table 4.1 except adjacency constraints. Moreover, samples are sorted based on circulation metric. In Figure 4.10, samples are sorted from left to right. The number of generated floor plans is 22.

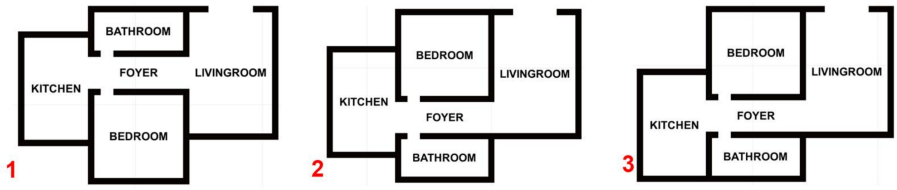


Figure 4.10: Samples of Scenario 4

Scenario 5. In this scenario, the input is exactly as specified in Table 4.1 . The number of generated floor plans is 5.

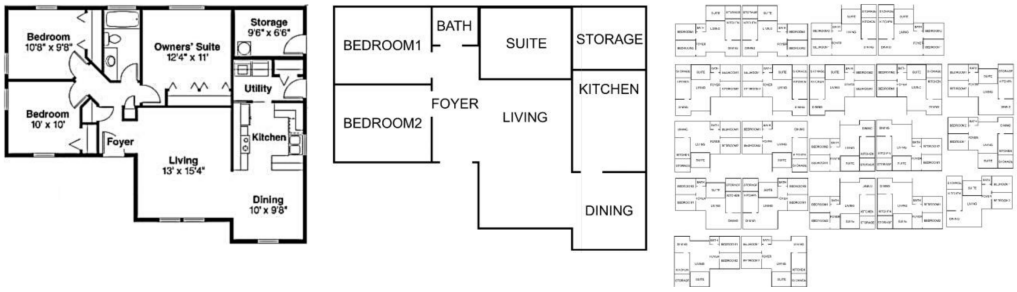
4.6 Experiment 2

In this experiment, we testify the efficiency of our system by generating a real world plan. Figure 11, (a) is a floor plan chosen from [Chatham, 2014] and (b) is one of the similar floor plan, among 22 generated floor plans, to the real world floor plan. In order to fill the input table of the system extract the dimensional and topological constraints from the source plan. For the sake of the clearness of this example, we had to limit the number of generated plans, therefore the flexibility parameters are set to be small, and also *Utility* and *Storage* are considered as direct neighbours. Moreover, the entrance is located on the south side of the *Foyer*. The generated plans are exactly the same as the source plan in terms of topological relationships, however, there are various arrangement of space units.

4.7 Conclusion

This paper attempts to assist architects by automating the process of floor plan generation by recommending feasible space layouts. Architects can specify some dimensional and topological constraints to reach their goals in a more precise manner. In experiment 1 we showed how more inputs produce fewer alternatives, and thus, an architect is more comfortable in choosing the best floor plan (considering the assumption that architects prefer to select the best plan from among a few number of recommendations). However, even by accurately determining the dimensional and topological constraints there are still infinite alternatives for a given problem. For instance, assume that an architect specifies that A and B should be neighbouring spaces, and also A should be above B (i.e. on a separate floor). Considering these requirements, there are still infinite number of alternatives satisfying the given constraints since the solution space of intersections are continuous .We believe that space layout planning has an interactive nature which requires human creativity to make the final decision; thus, the process of floor planning cannot be done in a fully automatic manner. However, in our system assists in this process to help save architect valuable time. The architect can interact with system by modifying the inputs and iterating the process of plan generation, if first results are not acceptable. Although in some situations our system cannot produce the final plan accurately, it gives the architect an abstract understanding of the possible arrangement of final plan. In this paper, floor plans are sorted based on the circulation quality metric. Choosing the best floor plan only based on one metric is not strong enough. Therefore as the future line of research, it would be highly interesting to measure the arch quality of a floor plan based on other factors such as privacy and lighting. Another substantial direction is to cover triangular and curve based

shapes.



(a) (b) (c)

Figure 4.11: (a) is the source plan, (b) one of the generated plans and (c) all the generated plans

Chapter 5

GENERATING ARBITRARY SHAPE LAYOUT BASED ON EVOLUTIONARY COMPUTING METHODS

Title	Optimizing Layout Using Spatial Quality Metrics and User Preferences
Authors	Arash Bahrehmand, Thomas Batard, Alun Evans, and Josep Blat
Journal	Computer and Graphics (submitted)
Year	2016
Keywords	layout planning, architectural modeling, genetic algorithm, personalization, interactive systems.

Optimizing Layout Using Spatial Quality Metrics and User Preferences

5.1 Abstract

Layout planning is a challenging phase of architectural design, which requires optimization across several conflicting criteria. We present an interactive layout solver that assists designers in layout planning by recommending personalized space arrangements based on architectural guidelines and user preferences. Initialized by the architects high-level requirements, an interactive evolutionary algorithm is used to converge on an ideal layout by exploring the space of potential solutions. We demonstrate the ability of our method to generate feasible floor plans which are satisfactory, based on spatial quality metrics and designers taste. The results show that the presented framework can measurably decrease planning complexity by producing layouts which exhibit characteristics of human-made design.

5.2 Introduction

Computational design is one of the most common tasks of immersive computer graphics projects, such as games, virtual reality and special effects [Lobos and Donath, 2010, Indraprastha and Shinozaki, 2012a]. Layout planning is a significant subcategory of computational design which has recently been the object of research in various layout configuration problems, such as the texture atlases, building arrangement in urban design, furniture arrangement, and interior design [Peng et al., 2014]. The design of 3D virtual environments aims to provide not only a visually engaging experience [4] but also a plausible understanding of the virtual entities by which the user can easily associate real world objects with corresponding 3D digital models [Vasin et al., 2015]. Therefore the digital artist should apply real-world design rules and architectural guidelines to create believable virtual buildings.

Layout design, which deals with the arrangement of elements in a space, is a fundamental aspect of the architectural design process. Due to the vagueness of the problem knowledge, a solution cannot be clearly formulated at the initial stage. Moreover, there is not a unique best solution for a design problem in architecture, and thus there is likely room to improve the quality of the designs obtained [Homayouni, 2000]. When formulated in those terms, the layout problems can be categorized as NP-complete [Maaroju, 2009], and managing the infinite space of

potential solutions requires significant effort towards obtaining optimal solutions. In these problems, one starts with an initial or current state, and progresses towards the desired state. In addition, given the contextual and subjective nature of solution quality assessment, the steps necessary for determining a solution can be vaguely defined, and lead to ill-structured solutions [Simon, 1977]. Although the actual best solution in the early stages of the design is far away, non-creative solutions of generations (stages) can be applied productively to generate more creative, more accurate solutions in new generations. Thus, we propose an evolutionary process to iterate toward the optimal solutions from an initial random state.

The demand for automatic design and creation of digital content, to be used in virtual environments, has been growing dramatically in the past few years [Ritchie et al., 2015]. However, creation of virtual layouts can be even more complex and time consuming for the digital artists than for the architects since, first, they normally do not have enough knowledge to address architectural requirements of the building, and secondly, unlike architectd who normally have to concentrate on a single layout at a time, the digital artist may need to create a mass of buildings for a particular district of a virtual city.

One of the tasks for designing interiors is to draft floor layouts, which currently, architects perform applying Computer Aid Design (CAD) techniques. The planning process, as any complex human activity, becomes more prone to error when the designer is faced with different levels of uncertainty in multi- dimensional problems. AI-based CAD tools have shown a potential to address the above mentioned issues by providing alternative design solutions, measuring the quality of each alternative and proposing design ideas based on data-driven approaches. Indeed, despite the growth of computational methods to generate and simulate floor plans, such methods are not quite satisfactory among designers yet. Our hypothesis is that this unsuitability of the solutions so far comes, on one side, from the fact that the formulation of the architects practices is weak and also that the methods could be improved by incorporating subjective aspects of the design in the processes.

This paper proposes an Interactive Layout Recommender System (ILRS) that provides floor layouts to the designer according to his/her preferences and architectural quality metrics. Our ILRS receives high level input constraints from the designer, generates an initial random set of layouts and performs an iterative process to improve the initial population. Our approach takes advantage of evolutionary computing to find solutions which satisfy a balanced account of input constraints. The objective function of our evolutionary algorithm takes into account architectural guidelines as well as users opinion to satisfy both functional and subjective aspects of the design. The proposed system contains an interactive 3D visualizer module that illustrates the 3D version of the generated 2D layouts, and provides the

designer with an interface to modify 3D architectural elements such as doors and windows. The target users of our system are digital artists (similarly to [Leblanc et al., 2011]) with a basic knowledge of architecture; however, the system can also benefit architects in early stages of the design process by providing various alternatives based on high level input requirements.

We evaluate different aspects of ILRS through a comparative analysis and three quantitative tests. The comparative analysis provides a detailed comparison of ILRS and three relevant works. The first quantitative test analyses the performance of our system through ten experiments based on five scenarios of increasing complexity. The second and third tests are validation experiments. The former is to demonstrate the ability of our system to provide novel, yet personalized, solutions by generating different versions of optimal solutions which fulfill similar architectural constraints; while the latter shows that our system can replicate human made floor plans by performing two experiments based on two different reference layouts.
In summary, our main contributions include:

- Handling concave and irregular convex shapes as input shapes and output layouts.

- Generating intelligently more accurate initial populations based on dense packing heuristics to decrease the computational costs and to direct the search into a particular region of the solution space which contains high quality solutions.

- Proposing a multi-parental recombination operator to improve the chance of attaining higher quality children that results in the faster convergence of optimal solutions.

- Applying users opinion in a fitness function in order to satisfy subjective aspects of the design.

- Supporting more spatial quality metrics such as circulation and privacy. Circulation means how well paths connect the space units. Privacy is based on the visibility of different points in the layout.

- Interactive 3D visualization of layouts that not only provides the designer with a better understanding of the current stage of the design but also allows him/her to choose his favorite styles of 3D architectural elements for a generated layout.

First the related work and background concepts are discussed before describing system overview (Sec 5.4), problem statement (Sec 5.5), optimization algorithm (Sec 5.6), implementation and interactive visualization (Sec 5.7), evaluation (Sec 5.8) and concluding the paper by discussion and limitations (Sec 5.9).

5.3 Related Work and Background Concepts

Layout planning is a substantial compound of several tasks involving various categories of information related to arrangement and construction of space units in a building [Karlen, 2011]. The designer first gathers information and data about building requirements to model design constraints, then attempts to arrange a layout which fulfills both architectural guidelines and required constraints [Rodrigues et al., 2013a]. Building layouts are mostly created as a result of an iterative trial-and-error process that needs substantial expertise and considerable amount of time [Merrell et al., 2010]. When increasing the number of design factors, space planning turns into a cumbersome task, which makes using specific software more reasonable and practical. In the past few decades several computer based systems were developed with the goal of assisting architectural designers at different stages of the design. Most of these systems apply a generative mechanism to create diverse solutions, and an assessment mechanism to measure the quality of the generated designs [Koenig and Knecht, 2014]. Comprehensive literature reviews of automatic layout planners can be found in [Rodrigues et al., 2013a, Koenig and Knecht, 2014]. The layout solvers have been applied in the following areas: Furniture planning [Merrell et al., 2011, Yu et al., 2011], residential space planning [Bahrehmand et al., 2014b, Merrell et al., 2011, Ansary and Shalaby, 2014, Peng et al., 2014, Bao et al., 2013, Knecht and Koenig, 2010, Homayouni, 2000, Koenig and Knecht, 2014, Koenig and Schneider, 2012, Merrell et al., 2010, Homayouni, 2007, Wong and Chan, 2009, El Ansary and Shalaby, 2014] and urban design [Peng et al., 2014, Elezkurtaj and Franck, 2001, Homayouni, 2000]. In the following subsections, we discuss this research categorized according to four criteria: Space unit shape, Synthesis method, Optimization model and Recommendation and personalization.

5.3.1 Space unit shape

Layout planning can be described as deciding on the best arrangement of geometric shapes (i.e. spaces) based on some architectural constraints. We categorise architectural layouts, in terms of the shape format, into three main groups:
Rectangular, when only rectangles are allowed as input shapes or output layouts.

Rectilinear, when only polygons whose edges meet at right angles, both convex and concave, are allowed as input shapes.

Arbitrary Polygon, when all polygon types (excluding curved shapes) are allowed as input shapes.

Although rectangular shapes have been widely applied in floor planning, rectilinear and arbitrary shapes still can be essential parts of a layout in a particular project. Especially when the site on which the building is going to be built is not rectangular, very likely some subspaces have to be deformed to a non-rectangular shape.

Many previous studies use predesigned shapes as input space units of the layout solver. Convex rectilinear shapes occur very often among other types of shapes in layout arrangement inputs [Tang and Wong, 2004]. A range of solutions have been developed for the synthesis with rectangles as basic units of the input shapes [Bahrehmand et al., 2014b, Müller et al., 2006, Merrell et al., 2010, Bao et al., 2013, Koenig and Schneider, 2012]. [Rodrigues et al., 2013a] presented a layout program, EPSAP, to solve a space allocation problem, which accepts only the rectangle as the shape type of inputs but it is able to generate layouts with rectilinear contour by combining the rectangles and decreasing the compactness level of the arrangement. [Bao et al., 2013] presented a building layout as a set of overlapping boxes which are parameterized by length and width attributes. The authors in [Merrell et al., 2010] try to achieve near-convex room shapes by penalizing large deviations from convexity with a shape cost function. According to them, one of the promising lines of research to improve the current state of art would be to add non-rectilinear and curve wall segments. Several studies have applied a top down approach by dividing a base rectangle into some subrectangles, which are interpreted as space units [Koenig and Knecht, 2014, Bahrehmand et al., 2014b, Knecht and Koenig, 2010]. [Bahrehmand et al., 2014b] specified some rectangles as null spaces after the subdivision stage in order to achieve non-rectangular contours for the final layout. In addition, sub rectangles are combined to generate L-shape spaces.
[Virirakis, 2003]] presented a bottom up problem solving method based on formal expressions to combine rectangular spaces, and the output of the system is a set of rectilinear floor plans which include rectilinear space units. Many studies have attempted to address the convex rectilinearity in the packing problem. However, few of them have focused on packing of convex and concave arbitrary polygons. Some papers discussed matching the arbitrary polygons as a general dense packing problem, while few of them tried to deal with this issue in architecture design [Lodi et al., 2002, Fujiyoshi and Murata, 2000]. Authors in [Chu and Young, 2004] state that using non-rectangular modules as pre-designed input shapes is not common in

practice, since non-rectangular shapes are generated as the result of combination of input rectangles in the optimization process; the vast majority of algorithms presented use orthogonal polygon shapes as space units in the process of generating layouts; and according to the same source, the use of non convex shapes is absent; [Rodrigues et al., 2013a] claims that most approaches used orthogonal polygon regular shapes as the input modules of the layouts.
In order to tackle the limitations of existing systems in terms of dimensional constraints, we augment the range of possible input shapes by supporting irregular polygons resulting in the increase of the diversity of the layouts geometry. Our proposed system both accepts deformable predesigned shapes as inputs, and is also able to generate new non-rectilinear shapes through deformation and combination of arbitrary polygons in the evolutionary process.

5.3.2 Synthesis method

In this section we discuss automatic layout solvers according to the method they use to arrange space units. [Koenig and Knecht, 2014]] classified the synthesis methods into two main groups: subdivision based and dense packing based. A *subdivision based method* divides a predefined shape, normally a rectangle, into subspaces or portions of the shape. [Bahrehmand et al., 2014b] divided an input rectangle into some rectangular spaces based on a binary tree technique where the tree contains the topological relationships between space units. Another subdivision method [Hahn et al., 2006] defines two types of regions in a building: temporary regions and built regions. The initial space consists of one temporary region; then, in each generation step, a temporary region will be divided into smaller temporary regions or turned into a new built region. [Marson and Musse, 2010] proposed a method for real time generation of layouts in virtual environments that applies Squarified Treemaps to present the hierarchical structure of subdivided areas. Moreover, some studies used Voronoi diagrams and Delaunay triangulation techniques to tessellate the input base shape [Coates et al., 2005, Harding and Derix, 2011]. The *dense packing method* arranges a number of spatial elements within a container without overlapping. In floor planning, the size of elements can be slightly modified during the packing process in order to fit space units with a minimal gap. Arvin and House [Arvin and House, 2002] demonstrated how physically based dense packing can be applied in space planning through modeling the design objectives as manipulation forces. [Elezkurtaj and Franck, 2001] used a similar concept as a solution for urban planning problems. In addition, [López-Camacho et al., 2013] presented an efficient algorithm for packing irregular polygons.
In our proposed system, we apply packing algorithms to arrange user desired input shapes, instead of subdivision based methods which limit the set of set inputs to

only those shapes that can be generated by the splitter algorithm.

5.3.3 Optimization model

The number of problems solved by optimization methods has been increasing. [Eiben and Smith, 2003] defined *global optimization* as a process of attempting to find the solution x^* out of a set of all possible solutions S which has the optimal value for some fitness function $\mathcal{F}$. However, most global optimization based techniques are not applicable in complex layout problems due to the very high number of solutions and contradictory nature of design factors. Randomized heuristics is another slightly different class of methods, that does not guarantee to find x^* as the global optimum, but rather as the best solution found so far [Eiben and Smith, 2003].

Considering the evolutionary structure of the real world design practice [Bentley and Wakefield, 1997], Evolutionary Algorithms (EA) have been used as the backbone of automatic space planning. Recently, Koenig and Schneider [Koenig and Schneider, 2012] applied an EA to generate viable layout solutions without feeding the system with extensive problem specifications. The method provides a non-overlapping arrangement of space units and their topological relationships. The combination of EA and stochastic methods is used not only to search within a very large solution space, but also to locally improve each individual layout solution in [Rodrigues et al., 2013a]. In [Wong and Chan, 2009], an EA based system can assist the layout planner by evolving the space topology in an efficient manner. Another hybrid EA technique that benefits from Genetic Algorithms (GAs) to locate and orient residential buildings optimally is used in [Ansary and Shalaby, 2014]. [Bao et al., 2013] used a Simulated Annealing method to generate initial candidate layouts while Merrell [Merrell et al., 2011] applied a similar stochastic optimization method to improve the quality of an individual one. Some approaches take advantage of machine learning techniques to organize the elements in a given area. For instance, [Indraprastha and Shinozaki, 2012a] presented a probabilistic model for arrangement of furniture that learns from a database of human made scenes. In another study [Merrell et al., 2010], an architectural program using a Bayesian Network trained on real world data is proposed.

Our system takes advantage of GAs, a subcategory of EAs; our algorithm evolves the quality of layouts based on some binary and unary varying techniques. In addition, a recursive attachment operator is applied to generate the populations of the proposed EA.

5.3.4 Recommendation and Personalization

Recommender systems are a subcategory of information retrieval systems that seek to predict the match between user preferences and item attributes [Naseri et al., 2015]. In the last couple of decades, recommender systems have experienced a dramatic growth in the number of users and the type of recommended items to try to solve the information overload. Although these systems have been mostly used in e-commerce and marketing, researchers have been attempting to apply recommendation techniques in other fields as well [Bustos et al., 2007]. A classic approach that has been applied in several recommender systems is to use item similarity to find the most appropriate item for the user. According to [Rafeh and Bahrehmand, 2012] user preferences might change over time, therefore the recommender system has to be capable of updating the list of users' favorite items. Automatic layout solvers, usually, take into account a set of architectural factors in the recommendation process.
Applying an interactive evaluation process in evolutionary algorithms already had some successful results as [Cardamone et al., 2011]. In order to incorporate users preferences in the search process, the system tries to find those layouts which have both good architectural qualities and a certain level of closeness to the users favorite choices. In the first generation of solutions, the contribution of this plan similarity factor is zero, since the user has not yet started to select his/her favorites. Over several iterations with user feedback, the system accumulates information regarding user preferences. Thus, in order to adapt the system to changes in users preferences, the newer layouts in the favorite list are weighted more heavily if the user gets better sense of what s/he wants over time with increasing iterations.
In our system, the user can update his/her favorite list at each generation by adding new layouts and removing those added in previous generations. Moreover, a layout arrangement can be presented in various styles based the on the culture of the site where that building is going to be built. ILRS provides the user with an interface to personalize layout appearance by choosing different styles for architectural elements of a 3D layout.

5.4 System Overview

Our system starts from the input set of dimensional and topological constraints, and outputs a set of floor plans through an interactive optimization process. Figure 5.1 illustrates an overview of the system, which consists of four main stages: getting input, generating initial population, evolutionary optimization and 3D interactive visualizer. In following a brief description of each stage is presented while Sec5.5 and Sec 5.6 provide a detailed description of the proposed framework

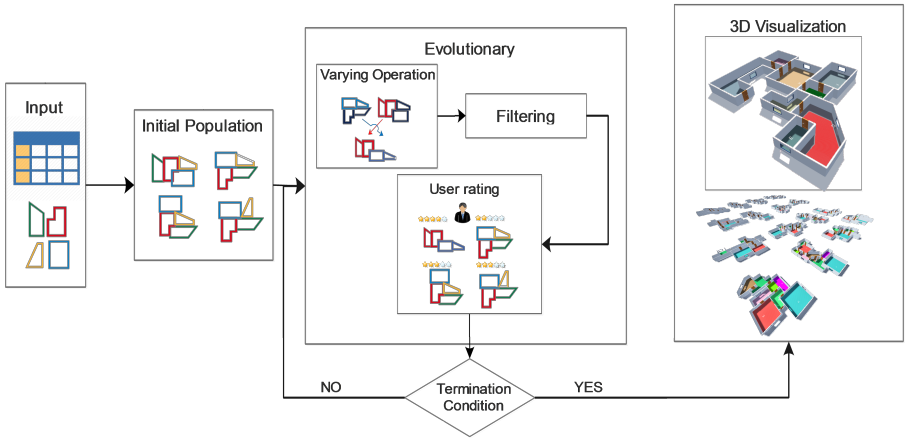


Figure 5.1: The general scheme of our proposed system (ILRS) that contains four main stages: **Input**, **Initial Population**, **Evolutionary** and **3D Visualization**. The user specifies input constraints at the input stage. In the second stage a semi-random initial population is generated. At the evolutionary stage the system updates the initial generation in an iterative process to reach the optimal solution. User may interrupt the system to rate layouts in some generations with the intention of conducting the search process. Once the termination condition is met, the system enters the final stage (3D visualization) in which the user can perform fine tune modifications to the resulting layouts.

along with mathematical formulation of the problem statement and optimization algorithm.

Input. In this stage the high-level requirements of the designer in the form of dimensional, topological, and opening constraints are organized. An example of input shapes in Figure 5.2 along with the table of topological, openings and area constraints in Table 5.1 are presented. The *adjacency* (column $\mathcal{M}$) value determines how important (preferable) is to the user the closeness of two space units. *adjacency = 1* between two space units, means they should (must) be neighbor(in all generated layouts), while *adjacency = -1* means the layout solver should avoid arranging these two space units next to each other, while -1 $<$ adjacency $<$1 determines the probability of being neighbor in a layout. The *area flexibility*, ϵ, determines how flexible is the space unit to get(be) deformed during the optimization. Moreover, the user is allowed to determine opening constraints through **W**, **D** and **E** columns.

Initial population. The initial population of the evolutionary stage is generated based on some heuristic methods. In order to generate a layout in the initial population stage, a copy set of input space units is created and each space unit in the set is randomly deformed based on area flexibility factor that is defined by user. Layouts in the initial population satisfy a set of constraints (see Sec 5.5.2). Each layout is created through semi-random attachment of space units.

Evolutionary. At this stage the system updates the initial population in an iterative process until the termination condition is met. Each iteration in the evolutionary optimization includes two main stages, generating offspring and filtering best layouts.
offspring are generated through recombination of parent layouts. In order to increase the productivity of varying methods, a parent selection method (see Sec 5.6.4) is defined that attempts to select the sub layouts set for the recombination that more likely generates high quality offspring.
In the filtering process layouts with the lower quality are removed from the population. The quality of a layout is measured based on a fitness function (see Sec5.6.4) that takes into account spatial quality metrics (architectural guidelines), topological relationship, overflow and user ratings (Sec 5.5.3). The set of spatial metrics includes: circulation, privacy, and compactness. Topological quality measures how close is the adjacencies of a generated layout to the desired adjacency relationships that are specified by the user in the input table. The overflow quality metric calculates how fit is the layout in the site boundary. The user may interrupt

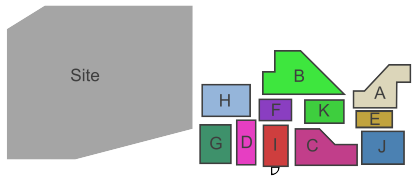


Figure 5.2: Input shapes of the first experimental analysis. The main entrance is defined in space unit *I*

the system at some generations to rate the layouts. User ratings contribute in measuring the quality of of layouts by giving more chance of survival to user's favorite solutions (see Sec 5.5.3). Layouts' rates at current generation are applied in the following generations as an indicator of new generated layouts' quality.

3D interactive visualizer. At this stage, the user is provided with some 3D interactive tools to visualize the layouts and modify(refine) 3D architectural elements. For example the user is allowed to convert add/remove s and change the style of architectural elements based on his/her taste (see Sec 5.7).

5.5 Problem Statement

Our proposed system computes qualified layout plans by maximizing over a space of valid layouts some (overflow, topological and architectural) quality functions and user's sanctification. The space of valid layouts is created according to the user's initial preferences, that is determined by the input stage. Knowing the user's initial taste (preferences) is fundamental in order for the algorithm to converge towards an accurate approximation of the solution of the optimization problem. It also helps us to design an initial condition for the algorithm that is not too far from the optimal solutions.

5.5.1 Problem Inputs

The input of the system is a set of m closed 2D polygons (the space units) $\mathcal{P}_1, \cdots, \mathcal{P}_m$ of $N_i, i = 1, \cdots, m$ vertices as well as a $m \times 6$ *input preferences* table (see Table 5.1). They indicate the initial preference of the user in terms of polygons shapes and deformation flexibility (first and third column, respectively), adjacencies (second column), and opening conditions (last three columns). It is worth nothing that for the table elements that user do not specify value, the system acts randomly. For

Space Unit	$\mathcal{M}$	ϵ	**W**	**D**	**E**
A	I(0.75), J(1.0)	0.25	*	J	
B	C(0.5), G(0.25)	0.20			
C	B(0.5)	0.25			
D		0.50	*		
E	J(0.75)	0.25		G	
F	K(0)	0.20	*		
G	B(0.25)	0.15		E	
H		0.25	*		
I	A(0.75)	0.50	*		*
J	A(1.0), E(0.75)	0.25		A	
K	F(0)	0.50			

Table 5.1: A sample of input preferences, where $\mathcal{M}$ denotes the adjacencies, ϵ is Area Flexibility, **W** is window **D** is Door and **E** is entrance.

instance, the system may add doors randomly between some adjacent space units that the user has not required the system to add door between them.
Adjacencies ($\mathcal{M}$ column) are encoded into a $m \times m$ matrix, $\mathcal{M}$, whose elements are defined as:

$$\begin{cases} 0 \leq \mathcal{M}_{ij} \leq 1 \\ \mathcal{M}_{ii} = 1 \\ \mathcal{M}_{ij} = \mathcal{M}_{ji} \end{cases} \tag{5.1}$$

where 0 means that the user does not want the polygons $\mathcal{P}_i$ and $\mathcal{P}_j$ to be connected, 1 means that he wants them to be connected, and 0.5 means that he does have any preference. Any value between 0 and 1 determines the user's level of interest to have the corresponding space units as neighbor.
The value ϵ_i for $i = 1, .., m$ determines how much the area of the shape $\mathcal{P}_i$ can be deformed during optimization.
Finally, W, D and E specify the preferences of the user about windows, doors and main entrance, respectively. The user can explicitly determine the existence of the doors between space units, windows on space units' walls and the main entrance on space units that are on the wall boundaries.

5.5.2 Problem Constraints

We require the layouts generated by the system to satisfy four constraints: geometrical, overlap, connectivity, and opening. Formally, we define a layout $\mathcal{L}_a$ as the union of a finite number of closed 2D polygons, and denote by $\mathcal{S}_\mathcal{L}$ the set of all possible layouts. We also denote by $\mathcal{S}_\mathcal{P}$ the set of arbitrary 2D closed polygons.

Geometrical constraints. The *geometrical constraints* $\phi_{\mathcal{G}e}$ are concerned with the likeliness of the polygons in a generated layout with the input space units (shapes that have been drawn by the user in the input stage) with respect to deformations tolerated by the user.
More precisely, given the m polygons $\mathcal{P}^1, ..., \mathcal{P}^m$ of input stage, and a layout $\mathcal{L}_a \in \mathcal{S}_\mathcal{L}$ such that $\mathcal{L}_a = \bigcup_{i=1}^{m} \mathcal{P}_a^i$, i.e., corresponding to $\mathcal{P}_a^i \in \mathcal{S}_\mathcal{P}$, we define:

$$\phi_{\mathcal{G}e}(\mathcal{L}_a) = \begin{cases} 0 \quad \text{if} \quad \dfrac{|Area(\mathcal{P}_a^i) - Area(\mathcal{P}^i)|}{Area(\mathcal{P}^i)} < \epsilon_i \qquad \forall i \\ \\ 1 \quad \text{otherwise} \end{cases}$$

where ϵ_i is the deformation flexibility of $\mathcal{P}_i$ determined by input preferences table. Notice that the geometrical constraint allows the polygons of $\mathcal{L}_a$ to have different numbers of vertices than the ones determined by the user.

Connectivity constraint. The *connectivity constraint*, $\phi_{\mathcal{C}v}$, is satisfied if the layout is a connected set, meaning that there should be a physical path between all possible space unit pairs. For instance, the layout in Fig. 5.3 is not connected since space units E and H do not have any door connecting them to the rest of the space units in the layout.
Overlap constraint. The *overlap constraint* $\phi_{\mathcal{O}v}$ avoids the area of the intersection between two space units of a layout to be too large, i.e.

$$\phi_{\mathcal{O}v}(\mathcal{L}_a) = \begin{cases} 0 \quad \text{if } Area(\mathcal{P}_a^i \bigcap \mathcal{P}_a^j) < \delta \ \ \forall i \neq j \\ 1 \quad \text{otherwise} \end{cases}$$

for some threshold δ determined by the system.

Opening. Three types of opening constraints are defined: *door*ϕ_D, *window* ϕ_W and *main entrance* ϕ_E.

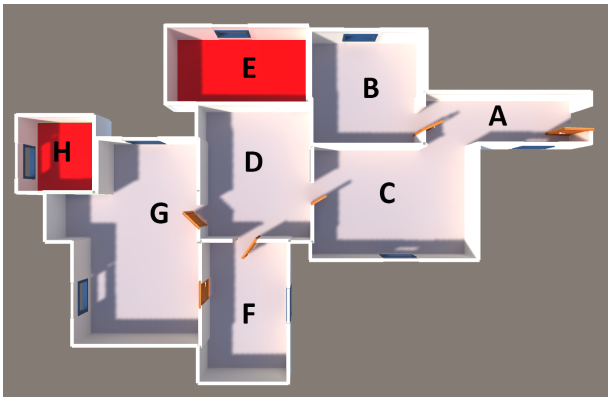


Figure 5.3: This layout is not connected since there is no opening to connect space units H and E to the rest of space units.

$$\phi_D(\mathcal{L}_a) = \begin{cases} 0 & \text{if} \quad \exists\, d \in \mathbf{D} \quad \text{between} \quad \mathcal{P}^i \text{ and } \mathcal{P}^j \Longrightarrow \mathcal{P}^i_a \cap \mathcal{P}^j_a \neq \emptyset \\ 1 & \text{otherwise} \end{cases}$$

meaning there should be a door between $\mathcal{P}^i_a$ and $\mathcal{P}^j_b$ if they are adjacent and the user has required the system (by column D at input table) to connect them by door d.

$$\phi_E(\mathcal{L}_a) = \begin{cases} 0 & \text{if} \quad \exists\, \mathbf{E} \text{ in } \mathcal{P}^i \implies \mathcal{P}^i_a \quad \text{is located at the boundary of } \mathcal{L}_a \\ 1 & \text{otherwise} \end{cases}$$

meaning if the space unit should that contains the main entrance (according to **E** column) is on the boundary of the layout, then the entrance constraint is satisfied by adding a door to one of its boundary edges.

$$\phi_W(\mathcal{L}_a) = \begin{cases} 0 & \text{if} \quad \exists\, \mathbf{W} \text{ in } \mathcal{P}^i \implies \exists\, \mathbf{W} \text{ in } \mathcal{P}^i_a \\ 1 & \text{otherwise} \end{cases}$$

meaning there should be a window on an edge of $\mathcal{P}^i_a$ if the user has required the system in the input table to have window for $\mathcal{P}^i$ and in overall, we define the opening constraint $\phi_{\mathcal{O}p}$ as

$$\phi_{\mathcal{O}p}(\mathcal{L}_a) = \begin{cases} 0 & \text{if} \quad \phi_D(\mathcal{L}_a) = \phi_E(\mathcal{L}_a) = \phi_W(\mathcal{L}_a) = 0 \\ 1 & \text{otherwise} \end{cases}$$

Attachment constraint. The system allows the user to manually attach some space units, before optimization stage, and present the result of attachment as a new constraint. Later, during the optimization process, the attachment constraint, $\phi_{\mathcal{A}tt}$ is fulfilled if the layout contains the predefined attachment between space

Figure 5.4: Overflows are highlighted with blue color.

units.

Finally, we combine the five (mentioned) constraints through the function $\phi_{\mathcal{H}}$ given by

$$\phi_{\mathcal{H}}(\mathcal{L}_a) = \begin{cases} 0 & \text{if} \quad \phi_{\mathcal{G}e}(\mathcal{L}_a) = \phi_{\mathcal{C}v}(\mathcal{L}_a) = \phi_{\mathcal{O}v}(\mathcal{L}_a) = \phi_{\mathcal{O}p}(\mathcal{L}_a) = \phi_{\mathcal{A}tt} = 0 \\ 1 & \text{otherwise} \end{cases}$$

from which we define the set of valid layouts $\mathcal{S}_{\mathcal{V}}$:

$$\mathcal{S}_{\mathcal{V}} := \{\mathcal{L}_a \in \mathcal{S}_{\mathcal{L}} \text{ s.t } \phi_{\mathcal{H}}(\mathcal{L}_a) = 0\}$$

5.5.3 Quality metrics

Quality metrics are $[0, 1]$-valued functions defined on the space of layouts. Four quality functions are defined: overflow quality, topological quality, spatial quality, and user rating.

Overflow quality function

The architect has to restrict the layout to the boundaries of the site where the layout is going to be built in. Figure 5.4 illustrates an example of the layout overflow from the site boundaries.
Our proposed layout solver attempts to avoid the overflow of the layout with respect to the site boundaries. To that purpose, we search for a rigid transformation of the layout (thus its shape is preserved) such that it does not exceed the site boundaries. However, finding the best fitting of an arbitrary shape inside another

one is not trivial. Besides, there might not exist a rigid transformation that makes the layout totally (fully) fits into the site. Hence, our proposal is to apply a heuristic method that is formulated as the search of the rigid transformation that minimizes the area of a given layout $\mathcal{L}_a$ that exceeds the boundaries of the given site Ω.
More likely, the optimal transformation is found by, first, mapping the barycenter $c(\mathcal{L}_a)$ of $\mathcal{L}_a$ to the one $c(\Omega)$ of Ω through a translation t, and then finding the rotation that minimizes the exceeding area. In other words, we search for the rotation $r^\star$ defined by

$$r^\star = \underset{r \in R_{t(c(\mathcal{L}_a))}}{\arg\min}\ Area((r \circ t(\mathcal{L}_a)) \setminus \Omega) \tag{5.2}$$

where $R_{t(c(\mathcal{L}_a))}$ denotes the set of affine rotations in $\mathbb{R}^2$ centered in $t(c(\mathcal{L}_a))$.
Then, the overflow quality function ϕ_{over} measures how much the boundaries of the layout exceeds the ones of the site, i.e.

$$\phi_{over}(\mathcal{L}_a) = \left(1 - \frac{Area(\mathcal{L}_a \setminus \Omega)}{Area(\mathcal{L}_a)}\right) \times \phi_{\mathcal{G}e}(\mathcal{L}_a \cap \Omega) \tag{5.3}$$

The term $\phi_{\mathcal{G}e}(\mathcal{L}_a \cap \Omega)$ constrains the inner part of the polygons in $\mathcal{L}_a$ that intersect the boundaries of Ω to be tolerable deformations of the user's preferred polygons (see Fig. 5.4).

Topological quality function

Given a layout $\mathcal{L}_a = \bigcup_{i=1}^{m} \mathcal{P}_a^i$, its adjacency matrix $\mathcal{M}^a$ is a $m \times m$ matrix that indicates for each polygon $\mathcal{P}_a^i$ the list of polygons that share at least one edge with it. It is the matrix defined by

$$\begin{cases} \mathcal{M}^a_{ij} = 1 & \text{if } \mathcal{P}_a^i \cap \mathcal{P}_a^j \neq \emptyset \\ \mathcal{M}^a_{ij} = 0 & \text{otherwise} \end{cases}$$

We define the topological quality $\phi_{topo}(\mathcal{L}_a)$ of $\mathcal{L}_a$ as a measure of the closeness between the adjacency matrix $\mathcal{M}^a$ of $\mathcal{L}_a$ and the adjacency matrix $\mathcal{M}$ (5.1) determined by the user in the input stage of the form

$$\phi_{topo}(\mathcal{L}_a) = \begin{cases} 0 & \text{if } \exists i \neq j \; s.t. \; \mathcal{M}_{ij} = 1 \; and \; \mathcal{M}^a_{ij} = 0 \\ 0 & \text{if } \exists i \neq j \; s.t. \; \mathcal{M}_{ij} = 0 \; and \; \mathcal{M}^a_{ij} = 1 \\ f & \text{otherwise} \end{cases} \tag{5.4}$$

where

$$f = 1 - \frac{1}{m(m-1)} \sqrt{\sum_{i,j=1}^{m} (\mathcal{M}_{ij}^{a} - \mathcal{M}_{ij})^2} \tag{5.5}$$

Although adjacency weights between space units are determined explicitly by the designer, still existence and position of the door (in the shared wall) between adjacent space units are considered as random parameters of optimization process for those that are not specified by the user explicitly in the table. In fact, two space units can be neighbor but not necessary connected by a door, however, the user can require the system to add door between two adjacent space units through the input preferences table (see Sect. 5.5.1).

Spatial quality metrics

In this section, we define a computational quality function ϕ_{arch} to evaluate the spatial quality of a layout based on some architectural guidelines such as circulation, privacy and compactness. More precisely, we define the function ϕ_{arch} as

$$\phi_{arch}(\mathcal{L}_a) = w_{\mathcal{C}}\mathcal{C}(\mathcal{L}_a) + w_{\mathcal{P}r}\mathcal{P}r(\mathcal{L}_a) + w_{\mathcal{C}p}\mathcal{C}p(\mathcal{L}_a) \tag{5.6}$$

s.t

$$w_{\mathcal{C}} + w_{\mathcal{P}r} + w_{\mathcal{C}p} = 1.$$

where $\mathcal{C}$, $\mathcal{P}r$, and $\mathcal{C}p$ are the circulation, privacy, and compactness quality functions respectively, and that are described below.

Circulation. A circulation path defines a semantic relationship between two spatial units in a floor plan. The circulation quality of a layout is measured based on the study that presented in [Bahrehmand et al., 2014a] which takes into account the following attributes for each path in the layout: path length, path complexity and traffic. However, in our system, we take into account paths connecting all possible points of space units instead of only considering the centers of space units (see Fig. 5.5).

Privacy. Privacy determines how and to what extend information about an individual or a group should be communicated to others [Westin, 1968]. Two types of privacy are defined in architecture: vocal privacy and visual privacy. In this paper, we only consider visual privacy which is closely tied to the concept of visibility. Visual privacy is related to a point p in an interior space being visible from an exterior space surrounding the interior one [Indraprastha and Shinozaki, 2012a]. The point p can be viewed from another point q if either p and q are in the same convex space unit or there is a direct path between p and q which crosses some

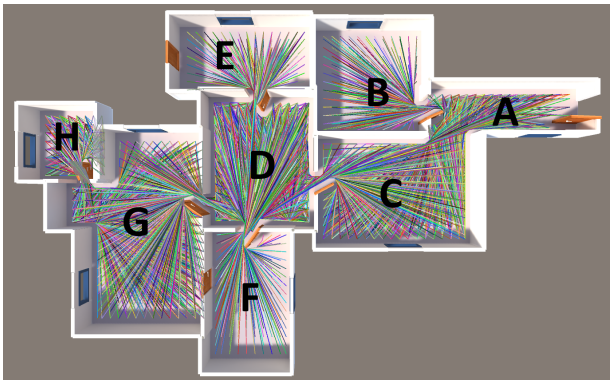


Figure 5.5: Visualization of paths connecting different points of the space units

windows or doors without any intersection with colliders. We apply a tessellation based approach to convert the continuous physical space of the floor layouts to a discrete space. The tessellation method creates a grid of cells that have the exact same shape and size. The physical space of a layout is tessellated to a finite number of non-overlapping cells and the privacy value (quality) is measured at the center of each cell, separately.
For a given point (cell) q in a space unit $\mathcal{P}^q$ inside a layout (floor plan) $\mathcal{L}_a$, we define three types of visibilities:

- v_q^o visibility from a point outside the floor plan. We call that space O.
- v_q^s visibility from a point inside $\mathcal{P}^q$.
- v_q^d visibility from a point in another space unit inside the floor plan.

Therefore, the (total) visibility of a point q is determined as:

$$v_q = (w_o v_q^o + w_n v_q^d + w_s v_q^s)/n_G$$

where

$$v_q^o = (\Sigma_{i \in O}\ \Theta_{i,q} \times \Delta_{i,q}/\Delta_{max})/n_o$$

$$v_q^s = (\Sigma_{i \in \mathcal{P}^q}\ \Theta_{i,q} \times \Delta_{i,q}/\Delta_{max})/n_s$$

$$v_q^d = (\Sigma_{i \in \mathcal{L}_a \setminus \mathcal{P}^q}\ \Theta_{i,q} \times \Delta_{i,q}/\Delta_{max})/n_d$$

$$\Theta_{i,q} = \begin{cases} 1 & i \text{ sees } q \\ 0 & \text{otherwise} \end{cases}$$

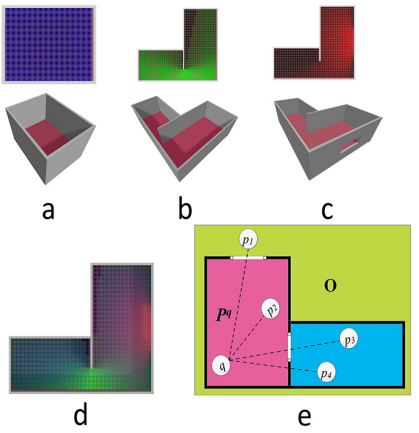


Figure 5.6: (a), (b) and (c) different levels of privacy are visualized. (d) is the visualization of the overall privacy for the layout in (c). In (e) point q in space unit $\mathcal{P}^q$ is been viewed from p_1(outside of the layout), p_2(inside the $\mathcal{P}^q$), p_3(inside the layout but not in $\mathcal{P}^q$). q and p_4 cannot see each other due to the existence of a wall collider between them.

$$n_G = n_o + n_s + n_d,\ w_o + w_n + w_s = 1$$

and n_G is the total number of cells, n_o the number of cells outside of the floor plan, n_s the number of cells inside $\mathcal{P}^q$, n_d the number of cells inside all space units of the floor plan except $\mathcal{P}^q$. $\Delta_{i,q}$ is the distance between cell i and cell q while Δ_{max} is the maximum possible distance between two arbitrary cells in the site; v_q ranges between 0 and 1.

In Figure 5.6 we visualize different types of visibilities of cells by which v_q^o is mapped to red, v_q^s is mapped to blue and v_q^d is mapped to green. Figure 5.6.a visualizes v_q^s for cells inside the space unit. Figure 5.6.b illustrates the visualization of v_q^d of all cells in space units. In Figure 5.6.c, a window is added which allows some cells to be viewed from outside. Thus, v_q^o is visualized for all the cells that can be viewed from outside. Figure 5.6.d illustrates the combination of all three types of visibility mapping.

The privacy quality of a point q is determined as:

$$\mathcal{P}r_q = 1 - v_q$$

Finally, the total privacy quality of the layout $\mathcal{L}_a$ is defined as the summation of the privacy quality of all points in $\mathcal{L}_a$:

$$\mathcal{P}r(\mathcal{L}_a) = \Sigma_{q \in \mathcal{L}_a} \mathcal{P}r_q$$

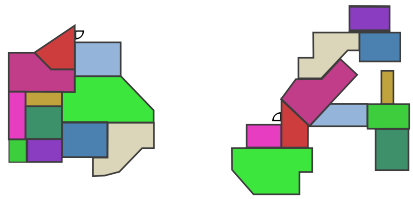


Figure 5.7: The right layout has a lower compactness in comparison to the left one.

Compactness. The compactness quality defines how efficient is a layout in terms of avoiding the creation of useless gaps between space units (see Fig. 5.7). Formally, the compactness level of a layout $\mathcal{L}_a$ that contains $m(> 1)$ space units is defined as:

$$\mathcal{C}p(\mathcal{L}_a) = \text{(Number of fit attachments)}/(m-1)$$

The notion of fit attachment is explained in details in Sect. 5.6.3.

User's rating function

Given a layout $\mathcal{L}_a$, the system allows the user to rate it through a score function $Rating$ such that $Rating(\mathcal{L}_a) \in \{1, 2, 3, 4, 5\}$, the score 5 indicates the user is fully satisfied by the layout whereas 1 means not at all.

5.5.4 Problem formulation

As mentioned above, we aim the system to generate the set S_{opt} of optimal valid layouts according to the overflow, topological and architectural quality functions, as well as the user's rating function. It can be formulated as follows

$$S_{opt} = \{\mathcal{L} \in \mathcal{S}_v; Rating(\mathcal{L}) = 5, \phi_{over}(\mathcal{L}) = \phi_{topo}(\mathcal{L}) = \phi_{arch}(\mathcal{L}) = 1\} \quad (5.7)$$

However, it is not guaranteed that such layouts exist. Moreover, even if the space S_{opt} is not empty, its construction can be very time consuming, which makes the problem (5.7) unrealistic.

Our proposal is then to design an algorithm that can generate a set of layouts $\widetilde{S_{opt}}$ approximating the optimal set S_{opt}, in the sense that the obtained layouts have high user's taste scores $Rating$ and high quality scores $\phi_{over}, \phi_{topo} and \phi_{arch}$.
To that purpose, we propose an optimization algorithm based on an evolutionary procedure such that the set of layouts generated at each iteration is, more likely,

better (in a sense that we will detail in Sect. 5.6) than the ones generated at the previous iteration. In particular, user feedbacks will guide the construction of the next iteration through the function $Rating$.

5.6 Optimization Algorithm

As discussed briefly in Section 5.4, the main goal of ILRS is to provide a set of optimal arrangements of input space units. The combination of EA and stochastic heuristics in [Rodrigues et al., 2013a] has proved its capability in generating feasible floor plans. In our framework, an optimization algorithm is applied in two main phases: generating semi-random solutions through heuristic techniques, and the evolution of solutions through EA algorithm. In this section, different components of the proposed optimization algorithm is presented through introducing the evolutionary scheme 5.6.1, user's relative taste function 5.6.2, attaching operator 5.6.3 and generating population 5.6.3.

5.6.1 The proposed optimization algorithm

The formulation of proposed evolutionary can be summarized in two stages (components):

1. The proposed evolutionary algorithm starts from a set $\widetilde{S^0_{opt}}$ of $\mu(>1)$ valid layouts (the initial population, whose construction is detailed in Sec 5.6.3). μ is the size of population and will be discussed in more details in Sect. 5.8.3.
Then, at each generation g, we are searching for the set $\widetilde{S^{g+1}_{opt}}$ of the μ valid layouts maximizing a fitness function $\mathcal{F}^g$, i.e.

$$\widetilde{\mathcal{S}^{g+1}_{opt}} = \underset{\substack{\mathcal{L}=\{\mathcal{L}_i\}\in(S_\mathcal{V})^\mu \\ \mathcal{L}\subset\widetilde{\mathcal{S}^g_{opt}}\cup Off^g}}{\arg\max} \sum_{i=1}^{\mu} \mathcal{F}^g(\mathcal{L}_i). \tag{5.8}$$

where Off^g denotes the offspring generated at the generation g (see Sect. 5.6.4 for more details about Off^g), $S_\mathcal{V}$ is the set of valid layouts and $\mathcal{L}_i$ is a layout from the population $\mathcal{L}$ at generation g . $\sum_{i=1}^{\mu} \mathcal{F}^g(\mathcal{L}_i)$ determines the total quality of a set of layouts. It is worth nothing that the total quality of layouts at generation g is not less than generation $g+1$.

Construction of the fitness function. Given the contextual and subjective nature of solution quality assessment in architectural design [Homayouni, 2000], we need

to take into account the user's preferences as a significant factor in the evaluation process. To that purpose, we allow the user to rate some layouts among $\widetilde{S^g_{opt}}$ at some generation g through the Rating function (described in Sect. 5.5.3). Since, over time, the user's understanding of the exact properties of the final goal can change, the system should be able to updates the value w^g_u whenever user rates new items. We set the value of w^g_u to zero in the first generation since the system has not received any feedback from the user. The user may interrupt the iterative process at some iterations to rate.
The more the user rates the selected layouts favorably, the more the fitness function of a given layout should take into account the user constraint rather than the architecture constraint. Therefore, we make the weight w^g_u be an increasing function of the rating, and we propose to make use of a function of the form

$$w^g_u = \sqrt[n]{\overline{Rate^g}} \tag{5.9}$$

where $\overline{Rate^g}$ determines the average of user's rates at generation g, for some $n \in \mathbb{N}$. Finally, as the ratings (the amount of user feedback the system has received so far) do not decrease throughout the generations, we have that the weight w^g_u is non-decreasing function of g.
We consider the fitness function $\mathcal{F}^g$ as a linear combination of the quality functions defined in Sect. 5.5.3 and a user's relative taste function ϕ^g_u (that we define in Sect. 5.6.2) of the form

$$\begin{aligned}\mathcal{F}^g(\mathcal{L}_a) =&(1 - w^g_u)\,[w_{over}\,\phi_{over}(\mathcal{L}_a) + w_{topo}\,\phi_{topo}(\mathcal{L}_a)\\ &+ w_{arch}\,\phi_{arch}(\mathcal{L}_a)] + w^g_u\,\phi^g_u(\mathcal{L}_a)\end{aligned} \tag{5.10}$$

where

$$w_{over} + w_{topo} + w_{arch} = 1 \qquad \text{and} \qquad 0 \leq w_{over}, w_{topo}, w_{arch}, w^g_u \leq 1$$

$\mathcal{F}^g$ is applied for all the layouts at each generation to find higher quality solutions and filtering low quality solutions. Thus, layouts at each generation are at least as good as the ones selected at the previous generations. Therefor it can be concluded that as the system runs more generations, the probability of surviving user's favorite layouts is increased. Ideally, at some generation $\underline{g}$, the weight w^g_u approaches(reaches) the value 1, more likely the user is fully satisfied by at least one of the layouts at the generation $\underline{g}$. However, in order to guarantee convergence of our evolutionary algorithm, we consider a maximal number of iterations g_{max}

for the iterative procedure (5.8), and we stop it after g^* iterations, where

$$g^* = \min(g_{max}, \underline{g})$$

2. The set $\widetilde{S_{opt}}$ approximating the optimal set S_{opt} (5.7) is the set of layouts having the highest fitness at the generation g^*, i.e.

$$\widetilde{S_{opt}} = \underset{\mathcal{L} \in \widetilde{S_{opt}^{g^*}}}{\arg\max}\ \mathcal{F}^{g^*}(\mathcal{L}) \tag{5.11}$$

$\widetilde{S_{opt}}$ contains at least one element and at the most μ elements. It is the proposed approximation of the optimal set S_{opt} (5.7) that we mentioned in Sect. 5.5.4.

5.6.2 The user's relative taste function

At the generation g, the user's relative taste function ϕ_u^g evaluates the quality of a layout in $\widetilde{S_{opt}^g} \cup Off_g$ based on the closeness level Sim (defined in formula (5.12)) of that layout to the user's rated layout at previous generation, S_{User}^g. To the best of our knowledge, none of the related work introduces any similarity metric to calculate the closeness level of two given layouts.
Let $\mathcal{P}^1, \cdots, \mathcal{P}^m$ be the m input polygons and $\mathcal{L}_a, \mathcal{L}_b$ be two layouts of the form $\mathcal{L}_a = \bigcup_{j=1}^m t_{aj}(\mathcal{P}^j)$, and $\mathcal{L}_b = \bigcup_{j=1}^m t_{bj}(\mathcal{P}^j)$ where t_{aj}, t_{bj} are deformations tolerated by the user. Assuming that the site Ω is a polygon, we define the similarity $Sim(\mathcal{L}_a, \mathcal{L}_b)$ between $\mathcal{L}_a$ and $\mathcal{L}_b$ as

$$Sim(\mathcal{L}_a, \mathcal{L}_b) = 1 - \frac{\frac{1}{m}\sqrt{\sum_{j=1}^m [c(t_{aj}(\mathcal{P}^j)) - c(t_{bj}(\mathcal{P}^j))]^2}}{\sqrt{(x_{max} - x_{min})^2 + (y_{max} - y_{min})^2}} \tag{5.12}$$

where $c(\mathcal{P})$ denotes the barycenter of the polygon $\mathcal{P}$, x_{max} resp. x_{min} denotes the maximum resp. the minimum among the x-coordinates of the vertices of Ω. In the same way, y_{max} resp. y_{min} denotes the maximum resp. the minimum among the y-coordinates of the vertices of Ω. The denominator in (5.12) guarantees that Sim is $[0, 1]$-valued.

Therefore we define the user's relative taste $\phi_u^g(\mathcal{L}_a)$ of a layout $\mathcal{L}_a \in \widetilde{S_{opt}^g} \cup Off_g$ as

$$\phi_u^g(\mathcal{L}_a) = (Rating(\mathcal{L}_{max})/5) \times Sim(\mathcal{L}_a, \mathcal{L}_{max})$$

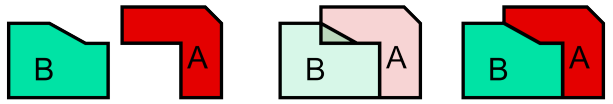


Figure 5.8: Removing the intersection region from A

where

$$\mathcal{L}_{max} = \underset{\mathcal{L} \in S^g_{User}}{\arg\max}\ Sim(\mathcal{L}_a, \mathcal{L})$$

is the most similar layout to $\mathcal{L}_a$ among the set S^g_{User} of user's rated layouts, and $Rating(\mathcal{L}_{max})$ determines the rating that has been assigned to $\mathcal{L}_{max}$ by the user.

5.6.3 Construction of a layout

The aim of this section is to introduce a recursive attachment operator that generates a semi-random valid layout from a given set of layouts. This operator is applied in the evolutionary process to construct the initial population $\widetilde{S^0_{opt}}$ from the input polygons $\mathcal{P}^1, \cdots, \mathcal{P}^m$ (considering a polygon as a layout with one single space unit), as well as the offspring Off^g at each generation g from the set $\widetilde{S^g_{opt}}$.

Attachment operator. Two given valid layouts are attached together by snapping two edges on their contours so that these layouts have at least one shared edge, partially or fully after applying the attachment. An *attachment operator* is defined as:

$$\begin{array}{rccc} \widehat{\cup}: & S_{\mathcal{L}} \times S_{\mathcal{L}} & \longrightarrow & S_{\mathcal{L}} \\ & (\mathcal{L}_a, \mathcal{L}_b) & \longmapsto & \mathcal{L}_a \bigcup t_{ab}(\mathcal{L}_b) \end{array}$$

where $t_{ab} \in T$ is a transformation that maps (snaps) a selected point $e^b_{n_b,k_b}$ on the edge $e^b_{n_b}$ of the contour $\mathcal{C}ntr(\mathcal{L}_b)$ of $\mathcal{L}_b$ to a selected point $e^a_{n_a,k_a}$ on the edge $e^a_{n_a}$ of the contour $\mathcal{C}ntr(\mathcal{L}_a)$ of $\mathcal{L}_a$. The transformation t_{ab} is the composition of the translation tr_{ab} of vector $e^a_{n_a,k_a} - e^b_{n_b,k_b}$ and the rotation r_{ab} determined by the angle between $e^a_{n_a}$ and $e^b_{n_b}$. We call the points $e^a_{n_a,k_a}$ and $e^b_{n_b,k_b}$ (which have the same coordinate after applying attachment operator) the *attachment points*.

From the attachment operator $\widehat{\cup}$ between two valid layouts, we derive a recursive

attachment operator $\widehat{\cup}^{k-1}$ between k valid layouts, as

$$\begin{array}{rccc} \widehat{\cup}^{k-1}: & (S_{\mathcal{L}})^k & \longrightarrow & S_{\mathcal{L}} \\ & (\mathcal{L}_1, \cdots, \mathcal{L}_k) & \longmapsto & \widehat{\cup}(\mathcal{L}_1, \cdots, \widehat{\cup}(\mathcal{L}_{k-1}, \mathcal{L}_k) \cdots) \end{array} \tag{5.13}$$

We say that an attachment between the k valid layouts $(\mathcal{L}_1, \cdots, \mathcal{L}_k)$ is *valid* if the output $\widehat{\cup}^{k-1}(\mathcal{L}_1, \cdots, \mathcal{L}_k)$ is a valid layout. Figure 5.10 illustrates examples of valid and invalid attachments.

Deformation. Layouts might be deformed before or after attachment. Each attachment operand is subjected to a random deformation before applying the attachment operator. The deformation is valid if it does not change the area more than a threshold (see Sec 5.5.2). The random deformation is performed by random translation of edges or vertices. However, in order to avoid very large translations, that more likely cause invalid deformations, we restrict the maximum amount of vertex movement (or edge) to a fraction of the operand (layout)'s area. In addition, the orientation of edge translation is perpendicular to the edge direction and the orientation of vertex translation is the direction of either of the edges that share the vertex.
Attachment may cause intersection between two layouts when at least one of them is convex. In our system, if applying attachment operator cause intersection, the intersected region is subtracted from one attachment operands. Figure 5.8 shows an example of removing intersection region after arrangement.

Semi-random edge selection. In our system, attachments between layouts $\mathcal{L}_b$ and $\mathcal{L}_a$ are not purely random. Indeed, we are searching for the *preferable* attachments based on input preferences table which are indicated through adjacencies column(see Table 5.1).
More precisely, given a random edge $e^b_{n_b}$ in $\mathcal{C}ntr(\mathcal{L}_b)$, that belongs to a unique space unit $t_{bj}(\mathcal{P}^j)$, we consider the ordered set of polygons $P^{j_1}, \cdots, P^{j_{k_j}}$ that have an edge in $\mathcal{C}ntr(\mathcal{L}_a)$ and whose adjacency with P^j is strictly greater than 0, i.e. such that $\mathcal{M}_{jj_1} > \cdots > \mathcal{M}_{jj_{k_j}} > 0$ where $\mathcal{M}$ is the input adjacency matrix. Then, given a random edge $e^a_{n_a}$ in $\mathcal{C}ntr(\mathcal{L}_a)$, that belongs to the space unit $t_{aj_1}(\mathcal{P}^{j_1})$, the system snaps $e^b_{n_b}$ to $e^a_{n_a}$. If this attachment is not valid, the system snaps $e^b_{n_b}$ to a random edge in $\mathcal{C}ntr(\mathcal{L}_a)$ that belongs to $t_{aj_2}(\mathcal{P}^{j_2})$, and so on until j_{k_j}.
Finally, if the attachment for j_{k_j} is not valid, the system snaps $e^b_{n_b}$ to a random edge in $\mathcal{C}ntr(\mathcal{L}_a)$.
Figure 5.9 shows an example of desired neighbors (determined by the user) for a

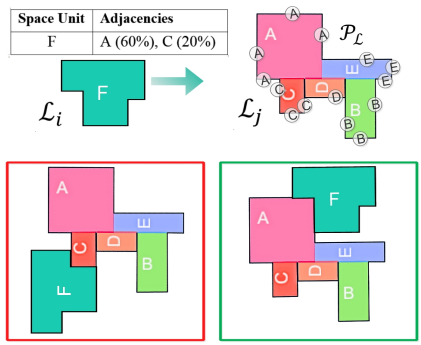


Figure 5.9: In the top row, the contour of $\mathcal{L}_j$ is labeled based on its space units names. The desired neighbors of F are A and C while the user tends to prefer A to B as the neighbor of F. In the second row, the left figure is not an appropriate attachment since F is attached to C while there is the possibility of a more appropriate ons by attaching F to A. The right figure is a preferable attachment.

space unit and a preferable attachment between this space unit, considered as a layout, and another layout.

Random point selection. In order to generate a valid layout, the attachment operator (5.13) is iterated several times until a valid attachment occurs. In order to improve the efficiency of the attachment process, a heuristic is applied in order to reduce the number of iterations by recording the history of previous *invalid* attachments.
More precisely, given two layouts $\mathcal{L}_a$ and $\mathcal{L}_b$, we define an attachment point pair $app = (app_1, app_2)$, where $app_1 \in Cntr(\mathcal{L}_a)$ and $app_2 \in Cntr(\mathcal{L}_b)$ as the pair of the two points that are determining an attachment $\widehat{\cup}$. At the iteration i, the proposed heuristic always attempts to pick a pair app^i that has a certain distance from the previous invalid pairs apps, that form a set called $\mathcal{I}^i$ and that contains $i-1$ invalid attachment point pairs. Therefore, we say that the pair app^i is *valid* if

$$\max(d(app^i_1, \mathcal{I}^i_{j,1}), d(app^i_2, \mathcal{I}^i_{j,2})) > t_{Dis} \quad \forall j \in \{1, \cdots, i-1\}$$

where d stands for the Euclidean distance in $\mathbb{R}^2$ and t_{Dis} is a threshold distance. Finally, we define

$$\mathcal{I}^{i+1} = \mathcal{I}^i \cup app^i$$

if the attachment between $\mathcal{L}_a$ and $\mathcal{L}_b$ determined by app^i is invalid.

For instance, in the Figure 5.10, t_{Dis} determines the radius of the dashed-circles, therefore any pair in which the two points are selected inside a circle is an *invalid* pair.

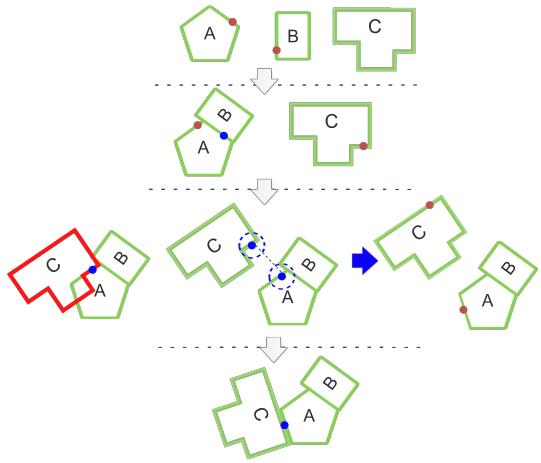


Figure 5.10: The first row illustrates the list of input polygons and the red dots indicate the attachment points. The second row shows a valid attachment between A and B. In the third row, the left layout is an invalid layout since it contains an invalid attachment between the contour of $\widehat{\bigcup}(A, B)$ and C while the right layout is a valid layout. At the next iteration, the points in the dashed circles receive less chances of being picked than others since an invalid attachment occurred on red dots.

Fit attachment. As indicated in Sect. 5.5.3 the compactness level of a layout relies on the number of *fit attachments*. We define a fit attachment between a layout $\mathcal{L}_a$ whose contour is convex and a layout $\mathcal{L}_b$ whose contour is concave as a valid attachment such that the attachment points are vertices $e^a_{n_a,0}$ and $e^b_{n_b,0}$ satisfying

$$\theta(e^a_{n_a,0}) = 2\pi - \theta(e^b_{n_b,0})$$

where θ denotes the intern angle between the two edges joining at a vertex.
For instance, both attachments in Fig. 5.9 are fit attachments.

5.6.4 Construction of the set $\widetilde{S}^{g+1}_{opt}$ in formula (5.8)

Given a set of individuals, EA algorithms attempt to increase the fitness of the population through two main operation: variation and selection. The former di-

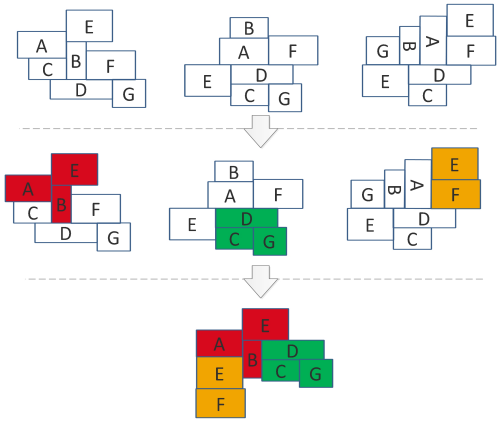


Figure 5.11: An offspring is generated as the consequence of combining three compatible sub layouts that are differentiated with red, green and orange colors.

verges the population to facilitate creativity and novelty, while the latter acts as a force to improve the quality of individuals. The variation operator generates new individuals from old ones through two types of methods: unary (mutation) and binary (recombination). The selection (filtering) operator distinguishes high quality solutions among all individuals in a population.
We apply the so called two-tier $(\mu + \lambda) - ES$ as a general form of evolutionary algorithm where μ is the number of individuals that are kept between generations (the sets $\widetilde{S^g_{opt}}$, $g \geq 0$ in formula (5.8)) and λ is the number of individuals that are generated through reproduction in each generation (the offspring set Off^g in formula (5.8)). In the last step, the system sorts the $\lambda + \mu$ layouts in the set $\widetilde{S^g_{opt}} \cup Off^g$ based on their quality scores $\mathcal{F}^g$, defined by formula (5.10), from which the set $\widetilde{S^{g+1}_{opt}}$ is generated.

Varying

Varying methods are representation dependent, meaning that the recombination and mutation should be defined based on the representation characteristics. In layout planning the varying methods rely on either the shape of space units or the topological relationships between space units. In our system, both shape and arrangement are variated. Each offspring, whether from recombination or mutation, is presented as a new solution in the solution space therefore a good representation of the varying operations guarantee the connectivity of the solution space. It means, the global optimum can be reached in a given sufficient time if the varying

operator has the capability of performing appropriate jumps in the search space.

Recombination is defined as a binary variation operator that merges information from parent layouts into one offspring [Togelius et al., 2015]. We define mutation as a particular type of recombination when parents are selected from the same layout. It randomly chooses a part of each parent for the mating process. By varying the seed value of the random function, several different children can be produced from two given parents. The resulting offspring is acceptable if it gives higher yields by inheriting desirable features of the parents. Therefore, parent characteristics have a substantial influence on the quality of the offspring which makes the process of parent selection so critical. Usually, in EA high quality individuals have a higher chance to become parent than low quality individuals, however the chances for low quality individuals should not be too small to avoid getting stuck in a local optimum [Eiben and Smith, 2003]. The proposed recombination method includes two main stages: selecting a set of parents, and creating an offspring as a consequence of combining these parents. However, selecting high quality parents does not fully guarantee the generation of a high quality offspring. We then introduce the concept of compatible parents in order to increase the chance of generating a high quality offspring. The aim of the next paragraph is to define rigorously a compatible set of layouts.

Compatibility of a set of layouts. Let $\mathcal{L}^g = \{\mathcal{L}_1^g, \cdots, \mathcal{L}_\mu^g\}$ be a set of μ layouts at the g-th generation. Recall that a layout $\mathcal{L}_i^g$, $i = 1, ...\mu$, in $\mathcal{L}^g$ is of the form

$$\mathcal{L}_i^g = \bigcup_{j=1}^{m} t_{ij}(\mathcal{P}^j)$$

for some deformations t_{ij} applied to the input polygons $\mathcal{P} = \{\mathcal{P}^1, \cdots, \mathcal{P}^m\}$.

As each layout $\mathcal{L}_i^g$ contains m space units, it possesses $2^m - 1$ subsets, called *sub layouts*. Then, a sub layout $l_{i,k}^g$ of $\mathcal{L}_i^g$, for $k = 1, ..., 2^m - 1$, can be written as

$$\overset{g}{}_{i,k} = \bigcup_{j=1}^{q_k} t_{ik_j}(\mathcal{P}^{k_j}) \tag{5.14}$$

for some $1 \leq q_k \leq m$, and $1 \leq k_1 < ... < k_{q_k} \leq m$.
We say that a sub layout is *valid* if it is connected.

Let $l^g = (l^g_{i_1,k_1}, ..., l^g_{i_N,k_N})$ be a set of N valid sub layouts, where

$$l^g_{i_n,k_n} = \bigcup_{j=1}^{q_{k_n}} t_{i_n k_{nj}}(\mathcal{P}^{k_{nj}})$$

for some $1 \leq q_{k_n} \leq m, 1 \leq k_{n_1} < ... < k_{n_{q_{k_n}}} \quad \forall n \in \{1, ...N\}$. The compatibility Cmp of the set l^g is 1 if

$$\bigcup_{n=1}^{N} \bigcup_{j=1}^{q_{k_n}} \mathcal{P}^{k_{nj}} = \mathcal{P} \text{ and } \mathcal{P}^{k_{n_1 j_1}} \neq \mathcal{P}^{k_{n_2 j_2}} \quad \forall (n_1, j_1) \neq (n_2, j_2) \tag{5.15}$$

and 0 otherwise, and we say that the set l^g is *compatible* if $Cmp(l^g) = 1$.

Finally, we say that a set $(\mathcal{L}^g_{i_1}, ..., \mathcal{L}^g_{i_N})$ of N layouts is *compatible* if there exists at least one set of sub layouts of the form $(l^g_{i_1,k_1}, \cdots, l^g_{i_N,k_N})$ that satisfies formula (5.15). Otherwise, the set of layouts is called *incompatible*.

Parent Selection. Some studies have applied 'intelligent' variation methods by incorporating problem knowledge in the parent selection process with the intention of increasing the chance of producing higher quality children [Brabazon et al., 2015]. We present an intelligent variation method that takes into account the quality and compatibility level of layouts in the process of parent selection. Moreover, in order to increase the chance of generating more creative layouts, our system supports n-ary parent set where $1 < n < m$ (m is the number of input space units). The parent set is the set of parent layouts that participates in the process of recombination while each parent layout contributes some of its space units in recombination (or final offspring). An appropriate varying method generates a (or some) new layout/s which are more likely to be of higher quality in comparison to their parents.
The fitness of an offspring does not only depend on the compatibility of the sub layouts that generate it but on the fitness $\mathcal{F}^l$ of each sub layout as well. Given a sub layout $l^g_{i,k}$, we define its fitness $\mathcal{F}^l(l^g_{i,k})$ as

$$\mathcal{F}^l(l^g_{i,k}) = w_{over}\, \phi_{over}(l^g_{i,k}) + w_{topo}\, \phi_{topo}(l^g_{i,k}) + w_{arch}\, \phi_{arch}(l^g_{i,k}) \tag{5.16}$$

In order to increase the probability of generating the λ children of highest quality, we construct a set $Cndt^g$ of λ *candidates* sets of compatible sub layouts $(Cndt^g_1, \cdots, Cndt^g_\lambda)$ having the highest fitness, where we define the fitness of

a set of sub layouts as the sum of the fitness of each sub layout, given by formula (5.16). The space $Cndt^g$ is given by

$$Cndt^g = \underset{\substack{S=(S_1,\cdots,S_\lambda)\\ S_i=(l^g_{i_1,k_{i_1}},\cdots,l^g_{i_{n_i},k_{i_{n_i}}})\\ Cmp(S_i)=1}}{\arg\max} \sum_{i=1}^{\lambda}\sum_{n=1}^{n_i} \mathcal{F}^l(l^g_{i_n,k_{i_n}}) \tag{5.17}$$

for $i_n \in \{1,\cdots,m\}$, and $1 \leq k_{i_1} < \cdots < k_{i_{n_i}} \leq 2^m - 1$.

Updating generation. For any candidate set $Cndt^g_i \in Cndt^g, i = 1,\cdots,\lambda$, we generate p valid layouts by attaching the sub layouts in $Cndt^g_i$ through q_i realizations, $q_i \geq p$, of the operator $\widehat{\cup}^{|Cndt^g_i|-1}$ defined in (5.13), and we select among these p layouts the one having the highest fitness (5.10), that we call $\mathcal{L}^{*g}_i$. Indeed, recall that an attachment is not necessarily valid, meaning that we have to perform more than p realizations of the attachment operator $\widehat{\cup}^{|Cndt^g_i|-1}$ if we want to obtain p valid attachments.
We then consider the offspring set Off^g at the generation g as the set containing these optimal layouts, i.e. $Off^g = (\mathcal{L}^{*g}_1,\cdots,\mathcal{L}^{*g}_\lambda)$ where

$$\mathcal{L}^{*g}_i = \underset{q_i \text{ realizations of } \widehat{\cup}^{|Cndt^g_i|-1}}{\arg\max} \mathcal{F}^g(\widehat{\cup}^{|Cndt^g_i|-1}(Cndt^g_i)) \tag{5.18}$$

Filtering(Selection)

After generating offspring, the filtering method is applied to select the best μ solutions. The proposed fitness function 5.10 is applied to rank the solutions. More precisely, the set $\widetilde{S^{g+1}_{opt}} = (\mathcal{L}^{g+1}_1,\cdots,\mathcal{L}^{g+1}_\mu)$ of parents layouts at the generation $g+1$ is determined by the set of the μ layouts among the parents and the offspring at the generation g having the highest fitness, i.e.

$$\widetilde{S^{g+1}_{opt}} = \underset{\substack{\mathcal{L}=(\mathcal{L}_1,\cdots,\mathcal{L}_\mu)\\ \mathcal{L}\subset \widetilde{S^g_{opt}}\cup Off^g}}{\arg\max} \sum_{k=1}^{\mu} \mathcal{F}^g(\mathcal{L}_k) \tag{5.19}$$

In Figure 5.12 we illustrate an example of the varying process and selection, including steps given by formulae (5.17), (5.18) and (5.19). For the sake of simplicity, the input polygons are considered as squares while the arrangement is defined by the horizontal order of squares. The last row in the figure shows the output of

the filtering method, i.e. the set $\widetilde{S_{opt}^{g+1}}$ at generation g which is the list of initial layouts for the next next generation $(g+1)$.

5.7 Implementation and Interactive Visualization

The outcome of generative design tools should be visualized to the artist in a near-real time fashion [Koenig and Schneider, 2012] with an acceptable appearance quality. Despite the wealth of intelligent design tools, many artists do not use these types of applications if they find a considerable difference between the computational drawing process and the conventional one [Kwon et al., 2009]. According to [Lobos and Donath, 2010, Indraprastha and Shinozaki, 2012a]immediacy, as a substantial characteristic of a generative tool, improves the reasoning quality in design process, while appropriate visualization of 3D interiors provides the artist with a better understanding of the current stage of the design. Designing a software application with the capability of visualizing high quality 3D scenes with the capability of real time interactivity is a traditional challenge in computer graphics. Our proposed framework provides the digital artist with the near-real time high quality visualization of the 3D layouts. Moreover, the user can modify the configurations of the visualizer based on his/her machine specifications to avoid low frame rates.

The implemented software application, with Unity 3D and C#on a machine with Core i7 CPU and 16GB RAM , is able to create the three dimensional version of a generated two dimensional floor layout. The 3D model is created based on some predefined architectural structures (elements):
Wall. The wall container consists of a mesh which represents the 3D wall object.
Door. A door container includes: left wall object, door object and right wall object.
Window. A 3D window consists of: left wall object, right wall object, top wall object, bottom wall object, window object.

Each structure consists of two components, transformation handler and object container. The former manages the transformation of the 3D structure in a layout while the latter contains a set of 3D meshes and textures which represents the user's desired style. Therefore, the user can simply add fine tune modifications to a layout by changing the properties of architectural elements.

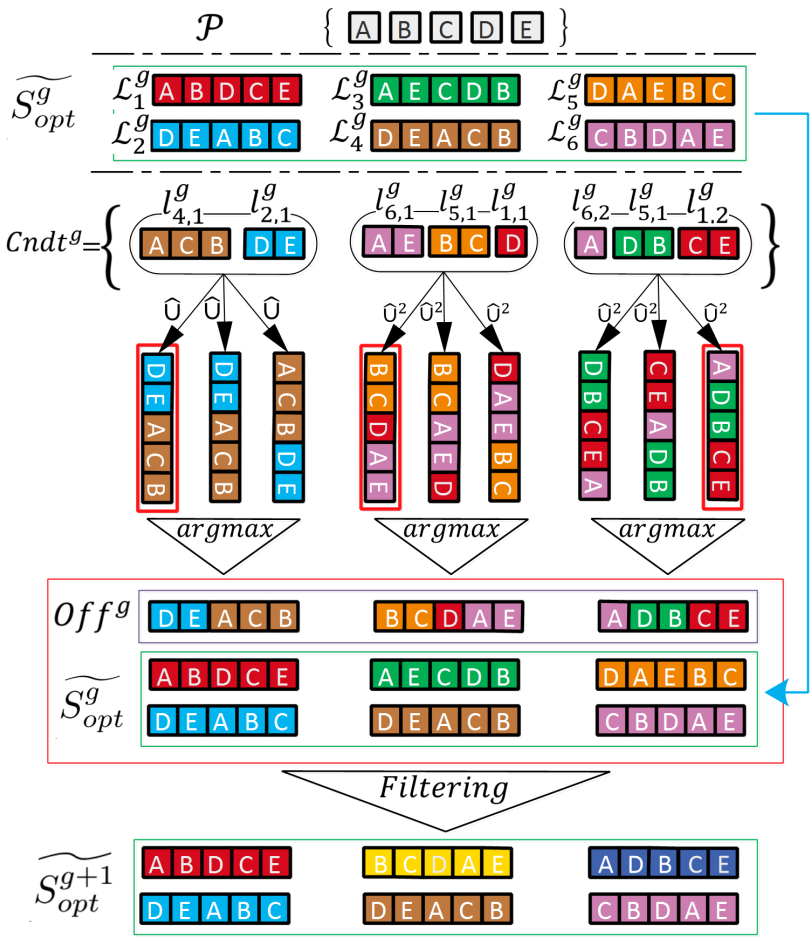


Figure 5.12: $\mathcal{P}$ is the set of input polygons and $\widetilde{S^g_{opt}}$ is the set of layouts in generation g. Each one of the three candidate sets generates three children by applying three times an attachment operator $\widehat{U}^k, k = 1, 2$. The best offspring of each candidate set is then added to the current list of layouts $\widetilde{S^g_{opt}}$. Finally, six layouts from $\widetilde{S^g_{opt}} \cup Off^g$ are selected for the $g+1$-th generation.

5.8 Evaluation

Automatic layout solvers are usually evaluated based on the quality, the variety and the validity of the layouts generated as an outcome. Additionally, the performance is measured in [Rodrigues et al., 2013b], based on the number of iterations the system needs to achieve the set of optimal solutions. The quality of the layouts is measured according to the architectural characteristics used in the optimization algorithm. The variety aims at demonstrating the capability to generate various alternatives. Validity intends to compare the artificial layouts generated with real world floor plans.
Our evaluation starts by the seldom performed comparison of the features of several relevant systems, including the one proposed in this paper. Then, we analyse quantitatively and in detail the performance, running a range of different experiments (10 different situations), with the system running in full automatic mode without user interaction, in order to understand raw performance. Next, we discuss validation and variety in detail in three different cases: a fully user driven experiment, and two cases where the goal is to validate the system against two pre-existing floor plans. Finally, we discuss and compare in detail our algorithmic approach with that of the system which is most similar to ours in the feature comparison: this helps us to assess more deeply the merits of each approach.

5.8.1 Comparative Analysis

Comparing the results presented in this paper with those of other research proposals is not possible, as the resulting systems cannot be reproduced, because implementation details are not publicly available, and the results provided in the papers cannot be directly compared to ours. It is likely that for this reason very seldom comparative evaluation has been made (and this includes the systems we discuss next). In this subsection we present a feature comparison with three studies, [Rodrigues et al., 2013b, Merrell et al., 2010, **?**]], which are most relevant to our approach. First, we provide an overview of these previous proposals, and an initial overall comparison.

EPSAP. Evolutionary Program for Space Allocation Problem (EPSAP) generates a set of floorplan layouts to be used by architects in early stages of design [Rodrigues et al., 2013a]. The input of the system contains a set of rectangles deformable within maximum and minimum areas, and some topological and opening constraints. The output is a set of rectilinear layouts. The method involves two stages: Stochastic Hill Climbing (SHC) and Evolutionary Strategy (ES). At the SHC stage the system generates a population of coherent layouts, starting by

a set of random rectangles, and generating new ones by applying a sequence of two types of random transformation operations. At the ES stage the fitness value of the generated layouts is improved iteratively until the termination condition is met. ES is implemented by performing *Wall Alignment* and *Void Remover*, and at each generation the fittest layouts are filtered and the remaining ones are replaced by new randomly generated layouts. The system is evaluated in terms of validation tests and performance analysis.

Computer-Generated Residential Building Layout (CGRBL) [Merrell et al., 2010]. This system provides an end-to-end approach for automatic generation of residential buildings in virtual environments. An architectural program is generated by training a Bayesian Network on a sample of real world architectural data. Then input, as a set of high level geometrical and topological requirements, is provided. The architectural program generates layouts fitting the requirements through a Simulated Annealing optimization method that considers some architectural quality metrics in its cost function. Finally the 3D exteriors and interiors corresponding to the 2D layouts are visualized as output. The system is evaluated through a validation analysis that measures the similarity of the layouts generated to some real world layouts selected from [Hanley Wood, 2007].

Recommendation of floor plans (RFP) [Bahrehmand et al., 2014b]. In this system, both a top-down method to subdivide a base rectangle based some input constraints and a bottom-up method to determine the dimension and ratio of subdivision are applied. The input of the system is a set of rectangles and topological constraints. As in EPSAP, the input rectangles are deformable. The interior non-rectangularity is supported via open walls while the exterior rectilinearity is generated by introducing the concept of null spaces, whose management is not intuitive. The generated layouts quality is measured based on circulation, and a validation test evaluates the system.

Table 5.2 reflects the features of the 3 systems and of our proposed one.

As the systems provide architectural design, one of the most important aspects is their support for architectural qualities. In this respect, ILRS and EPSAP are the systems which support more quality constraints, with ILRS supporting circulation and privacy beyond EPSAP. Only ILRS and RFP support attachment constraints meaning that essential users requirements are included from the initial stage, allowing to filter out irrelevant arrangements from the space of solutions, and thus saving computation time.

Another important architectural feature is the shape variety allowed. ILRS is the

	ILRS	**EPSAP** [Rodrigues et al., 2013b]	**CGRBL** [Merrell et al., 2010]	**REP** [Bahrehmand et al., 2014b]
Quality Constraints	Sp Ov, Cr, Op Sp, Pr, Cp, Cn, LD, and Of	Sp Ov, Op Or, Op Ov, Cp, Cn, LD, and Of	Sp Ov , Cn, LD and Of	Sp Ov, Cn, LD, Op Or, and Of
Attachment Constraint	Supported	Not Supported	Not Supported	Supported
Shape Variety	Arbitrary Polygon	Rectilinear Polygons	Rectilinear Polygons	Rectilinear Polygons
Interaction before Optimization	Specify Input requirements	Specify Input requirements	Specify Input requirements	Specify Input requirements
Interaction during Optimization	Rating Layouts	No Interaction	No Interaction	No Interaction
Interaction after Optimization	Fine tune modifications	No Interaction	No Interaction	No Interaction
3D Visualization	Supported	Not Supported	Supported	Not Supported

Table 5.2: Features Comparison. SO(Space Overlap), Op Or(Opening Orientation), Op Ov(Opening Overlap), Op Sp(Opening Space Unit), Cp(Compactness), Cn(Connectivity), LD(Layout Dimension), Of(Overflow), Cr(Circulation) and Pr(Privacy)

only system allowing for arbitrary polygons, which means an important flexibility. In the subsubsection 5.8.4 we discuss in detail the problem formulation of both ILRS and EPSAP which leads to this different variety.
Both ILRS and RFP lead to 3D representations of the results, which make them more applicable in Computer Graphics, for instance in the procedural generation of 3D layouts in video games.
All the methods support the initial input requirements provided by users. However, ILRS is the only one where the designer can interact during the layout generation process to allow the system to guide the search process towards layouts that match better the users tastes. Additionally, ILRS supports the fine tuning of the final results, which makes it more applicable. The proposed system has a full interactivity range: it can be fully automatic if the user only adds input constraints, or is fully interactive as at each generation s/he can rate and modify all architectural elements through 3D interactive tools.

5.8.2 Performance Test: the system in automatic mode

The performance tests contemplated five scenarios, each with different geometrical and topological constraints; on the other hand, they are of increasing size. For each scenario, we ran the system with two different population sizes as, hopefully, with larger population sizes the termination condition is reached earlier, despite each stage requires more computing time. Thus, we ran 10 experiments altogether, and with them we intended to estimate the performance parameters of the system, its suitability to deal with a variety of conditions, and how scalable our approach was. In these experiments we set $w_u = 0$ in the fitness function, to discard the contribution from user interaction: we are testing the performance as a fully automatic system. The termination condition is met when the overall quality of the best layout does not change more than a threshold over the last four generations (as in [Rodrigues et al., 2013b]).
Table 5.4 reflects the parameters of the different scenarios: $|\mathcal{P}|$ is the number of space units, μ is the population size, and *Topo* represents the density of adjacency matrix. The table also shows the results we computed:

- The average time, $t(\mathcal{L}_i)$, to generate a single valid layout in the semi-random generation of valid layouts.
- The average time spent on each generation, $t(g_n)$, creating layouts through recombination, and selecting the fittest layouts.
- The number of generations g^*

μ	λ	$itrp$	g^*	$\|\mathcal{P}\|$	w_{arch}	$w_{\mathcal{C}p}$	$w_{\mathcal{C}}$	$w_{\mathcal{P}r}$	w_{over}	w_{topo}
32	16	9	86	11	0.5	0.4	0.4	0.2	0.25	0.25

Table 5.3: Constraints values of quantitative analysis

Figures 5.13, 5.14, 5.15, 5.16, 5.17, and 5.17 show a screenshot of an optimal layout of the last generation of each scenario and the graphs of the anytime behavior [Boddy and Dean, 1994] of our EA. In the left image of each figure, the black contour determines the geometry of the site boundary.
According to the results in the table and in the figures:

1. The anytime behavior graphs demonstrate that our EA system improves the overall quality of initial populations in a sublinear way and reaches the optimal solutions after a certain number of generations.

2. The screenshots of optimal layouts of the last generation of each scenario, and especially, of that with 64 space units, demonstrate that the system can address in a robust and scalable way a variety of scenarios.

3. The experiments indicate that the proposed system can generate various layout solutions.

4. The average time to create a valid layout depends on the number of space units and the density of the topological graph.

5. As expected, increasing the number of layouts leads to increase the average time of a generation.

6. The average time of a generation increases superlinearly with respect to the number of layouts, as the larger this is, the heavier the required computation of the techniques including parent selection and recombination.

7. Increasing the population size seems to lead to reach the optimal layouts earlier.

5.8.3 Competence and Validation Analyses: 3 cases with a high level of user intervention

In this subsection we discuss 3 experiments. The first one illustrates a full use of our system with a high user intervention; and we discuss its capability to assist users (ideally, digital artists) to reach creative and yet valid layouts. The second and third experiments validate the system in real world situations, namely,

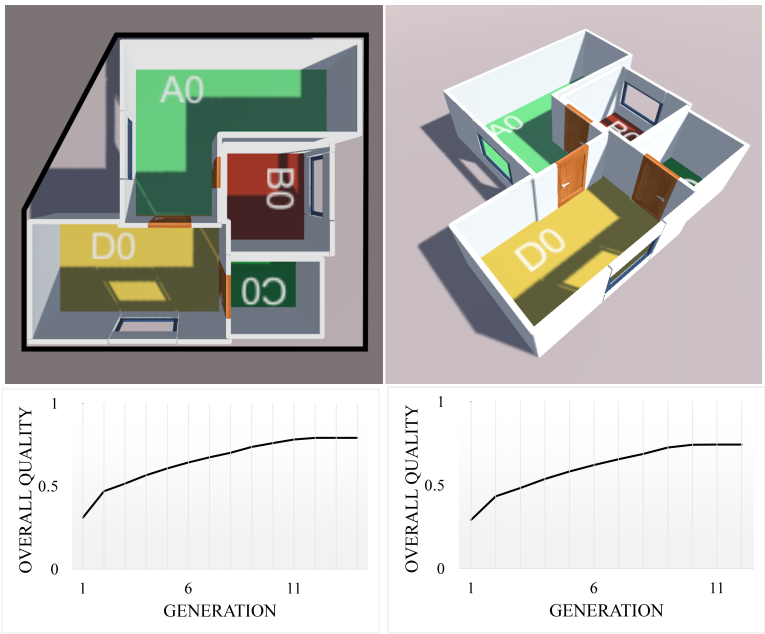


Figure 5.13: First scenario of performance test

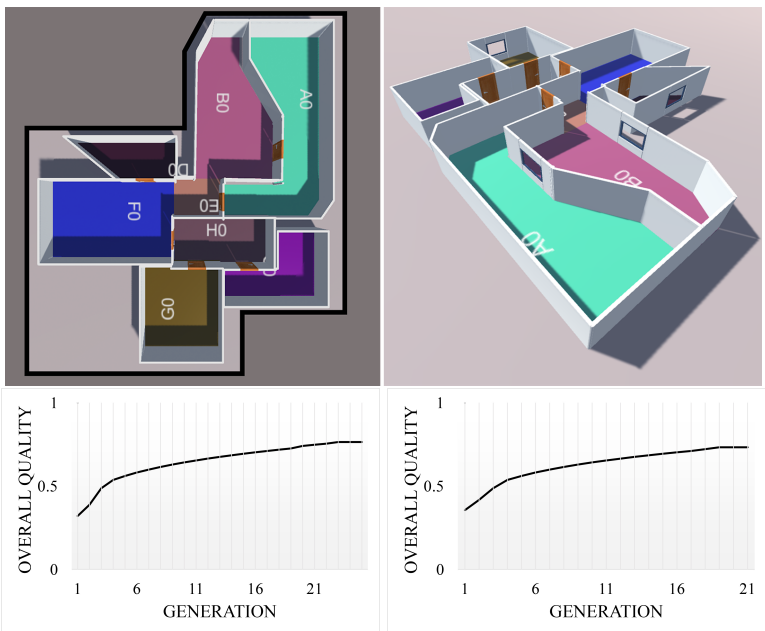


Figure 5.14: Second scenario of performance test

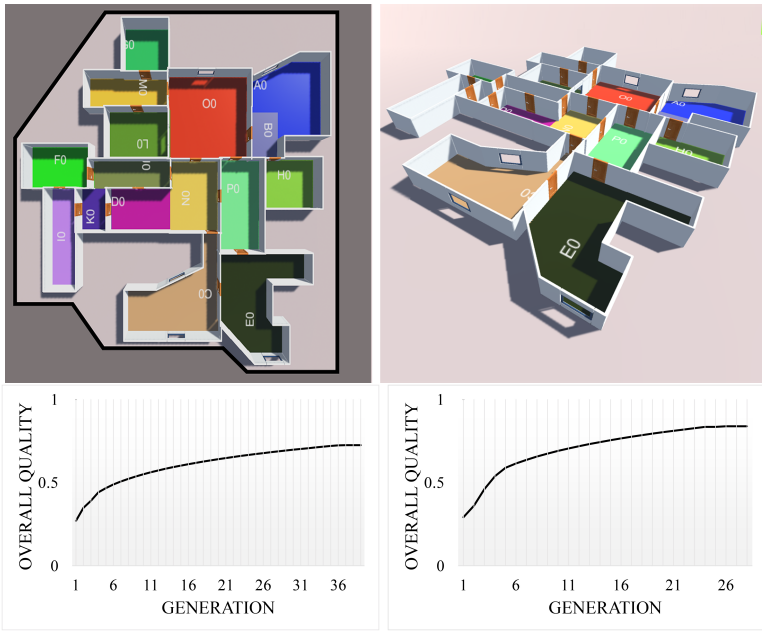


Figure 5.15: Third scenario of performance test

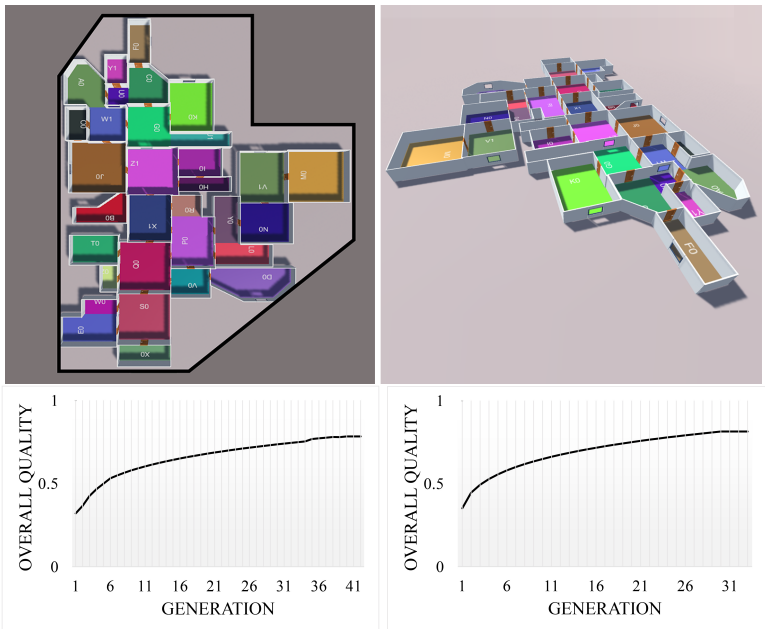


Figure 5.16: Fourth scenario of performance test

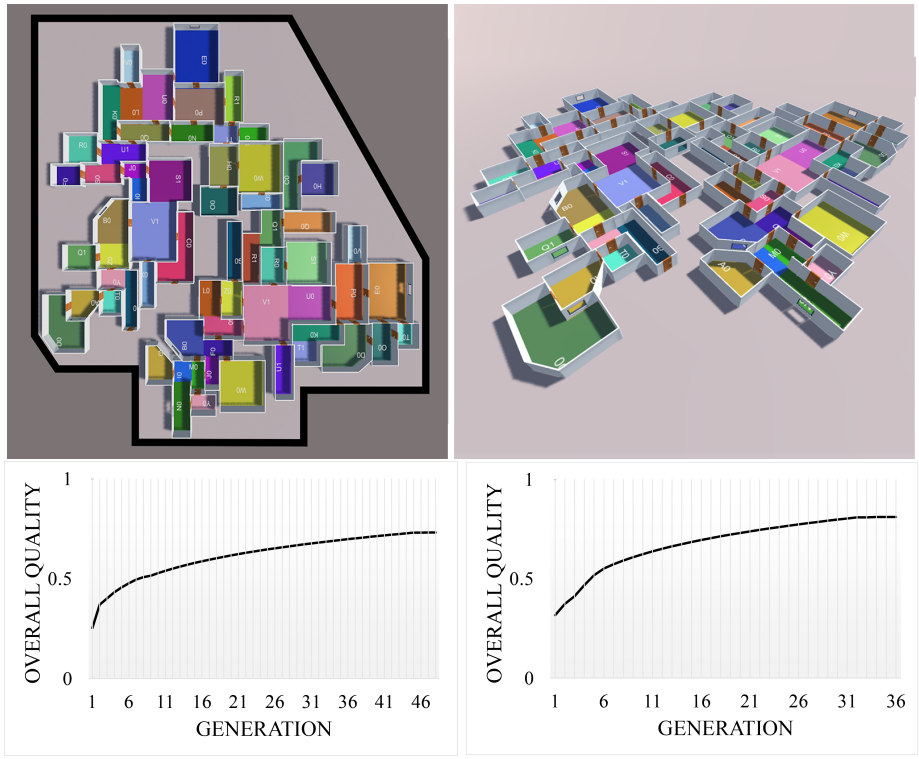


Figure 5.17: Fifth scenario of performance test

comparing the results obtained by the system with two real world floor plans which are defined as goals, a strategy previously followed to measure the competence of an automatic layout solver [Merrell and Schkufza, 2008, Rodrigues et al., 2013a, Bahrehmand et al., 2014b].

An illustrative use case in detail: Competence

The experiment was performed based on some input shapes, shown in Figure 5.2; it can be seen that the user specified the preferable location of windows and main entrance. Topological constraints (adjacency matrices), area flexibility, windows and doors were specified in the way represented in Table 5.1. The intended goal was to find layouts that were compact and provided a good level of circulation. Therefore, the weights of compactness $w_{\mathcal{C}p}$ and circulation $w_{\mathcal{C}}$ were set to be larger than those corresponding to other factors (some other possible settings are discussed in section 5.9). This is shown on table 5.3, which indicates other choices to run the experiment: μ was based on the performance results of the automatic system pre-

Scenario	$\|\mathcal{P}\|$	μ	Topo	$t(\mathcal{L}_i)$	$t(g_n)$	g^*
1	4	16	∼30%	0.43	6.71	10
		32	∼50%	0.67	21.01	8
2	8	16	∼20%	2.18	33.31	25
		32	∼40%	3.34	68.11	21
3	16	32	∼20%	4.89	89.74	39
		64	∼30%	6.53	423.51	28
4	32	32	∼15%	29.61	766.96	42
		128	∼20%	48.22	7622.18	33
5	64	128	∼10%	98.05	19377.48	48
		256	∼20%	133.10	26351.14	36

Table 5.4: Scenarios and results of the performance tests; time in seconds

viously discussed; and with μ between $|\mathcal{P}|$ and $2|\mathcal{P}|$, we expected to find optimal solutions in less than $|\mathcal{P}|^2$ generations; however, the variation can be large due to the randomness of the generative process, reinforced if the system present layouts that are not matched with the user opinion in the first generations; g_{max} is the total number of generations and $itrp$ is the number of rating interruptions by the user. The termination condition was met when the user assigned five stars rating to at least one layout.
The optimal layout was found in generation 86 and the average running time of each generation was about 189 seconds. While this time is only indicative, as the duration time can vary depending on shapes complexity, input requirements, etc. (as discussed in subsection 5.8.2), it represents approximately 4.5 hours to get a good layout, a task that might take an architect about a week. A more detailed discussion of the experiment is based on the graphs presented in Figure 5.18, and the visual results in Figure 5.19.

Figure 5.18 shows the graphs representing the values of the different quality metrics (Y axis) versus the generation number (X axis). We see that:

1. The average quality is increasing at each generation and gets closer to the highest quality layout by approaching g^*: this means that the system provides more high quality layouts at each generation.

2. The graphs, except for W_u, have sublinear growth, and roughly logarithmic shape.

Within Figure 5.19, Figure 5.19a is a screenshot of nine solutions at the first generation; Figure 5.19b shows some layouts of the last generation, and layout 6 (high-

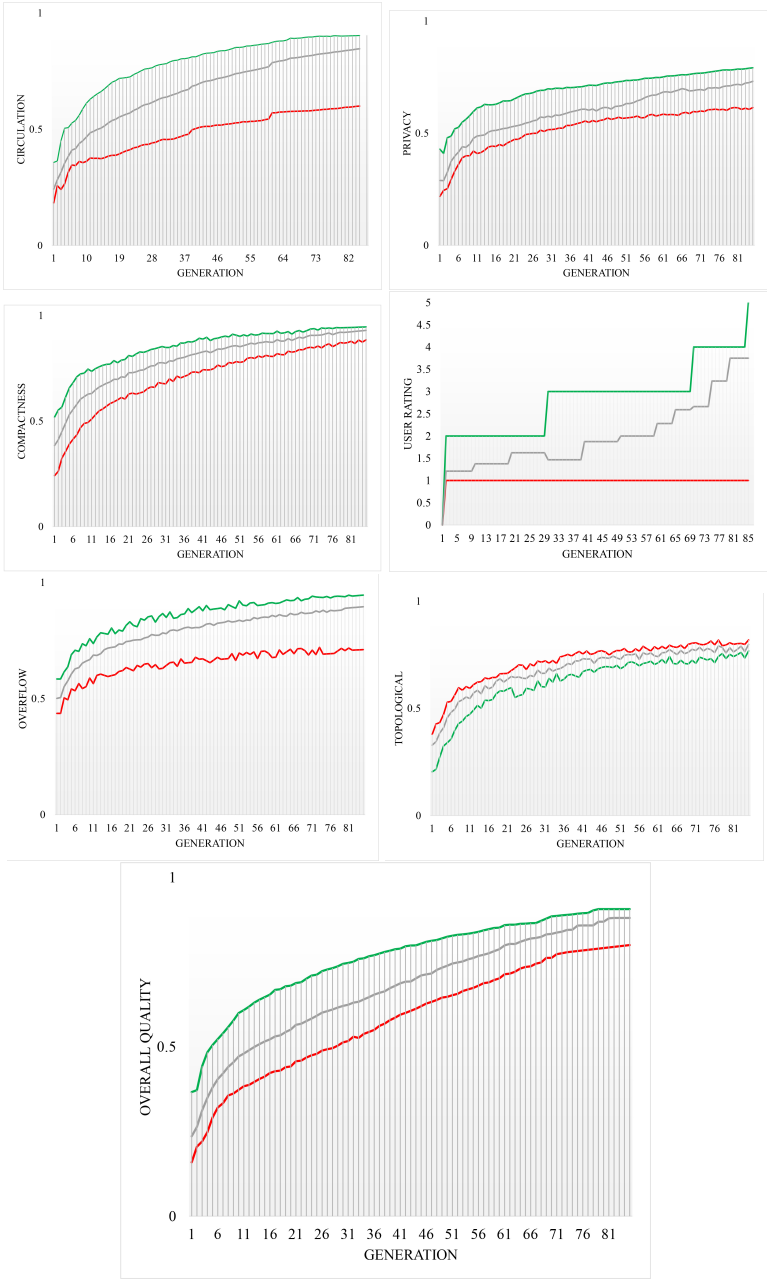


Figure 5.18: Quality statistics of the competence experiment

lighted by a red contour) was selected as the most optimal layout; it is presented with the 3D visualizer in Figure 5.19c. We can remark that:

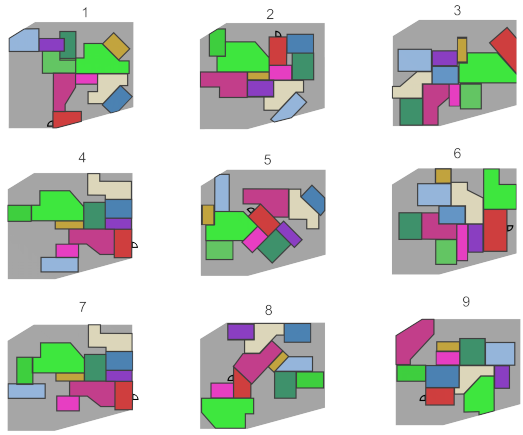
1. The space units *I* and A have been deformed most in comparison to other space units; while the deformations of units H and K observed in layout 9 are examples resulting of satisfying site boundary constraints.

2. As illustrated in Figure 5.20 the most optimal layout is the offspring of three parents, L4, L5 and L2.

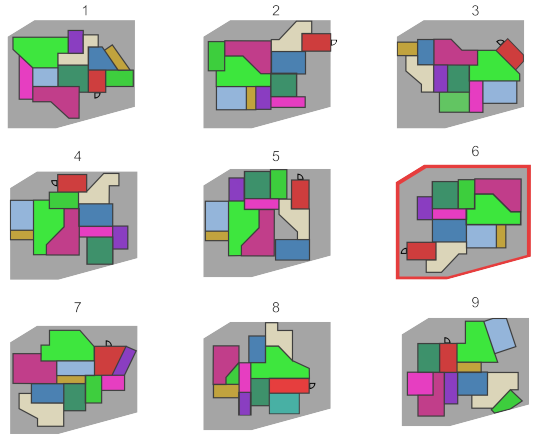
This experiment illustrates that the proposed system can generate various layout solutions where the average quality of individuals in each generation improves until the termination stage. Our results (for instance, the minimum number of generations until termination) are difficult to compare with those of other related systems as our proposed method involves user's opinion in the evaluation of intermediate solutions [Cardamone et al., 2011].

Validation Analysis against real world examples

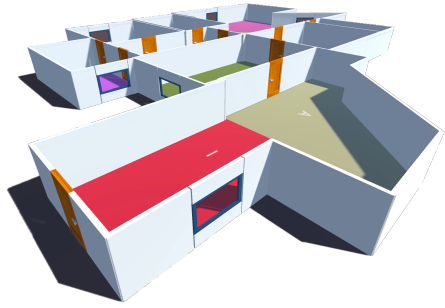
To validate our approach against real world situations, we ran two different experimens, with actual floor plans (See Figure 5.21) set as targets. The first reference layout was Kaufman House plan, which has been used in several studies [Indraprastha and Shinozaki, 2012a] for computational analysis; while the second was selected from [house designers., 2016].
We defined the respective input constraints based on the specifications of the floor plans, and setup the system to generate layouts similar to the reference layouts. However, we did not require the system to fulfill all the topological constraints, with the intention of increasing the variety of alternative solutions. In these experiments, as the user knows exactly the layout which should be achieved and looks for similarity, w_u is larger than other weights in the optimization process. Interactive based methods, such as our approach, involve users' preferences in the process of evaluating the solutions [Hastings et al., 2009, Martínez and Yannakakis, 2011, Shaker et al., 2013, Yannakakis and Hallam, 2007], and they usually provide a reliable and accurate estimator of the user's taste [Togelius et al., 2015]. These methods require occasional interruption of the iterative process. In the first experiment the user conducted the search process by stopping the system 4 times for rating, while the system was interrupted 6 times in the second experiment.
Figure 5.22(left) and Figure 5.22(right) show nine layouts of the last generations. The layouts most similar to the respective reference are highlighted with a red rectangle. They demonstrate that ILRS can replicate floor plans made by a human designer. The layouts highlighted in blue rectangle have the same architectural



(a) an screen shot of nine layouts of first generation.



(b) an screen shot of nine layouts of 86th generation.



(c) The 3D version of one the layouts in the 86th generation.

Figure 5.19: Some results of *Alternative layouts creation* experiment.

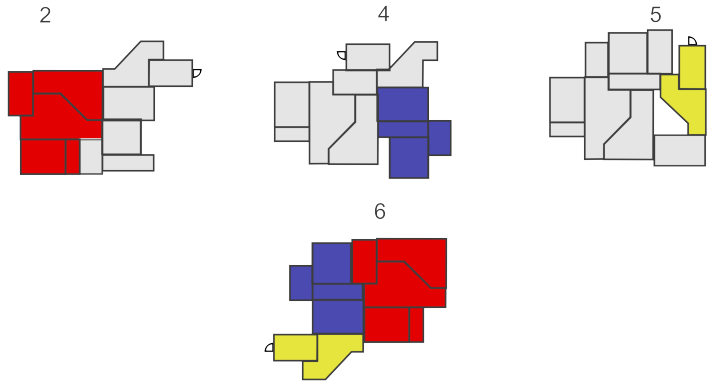


Figure 5.20: The layout 6 is the consequence of the recombination of L4,L5 and L2

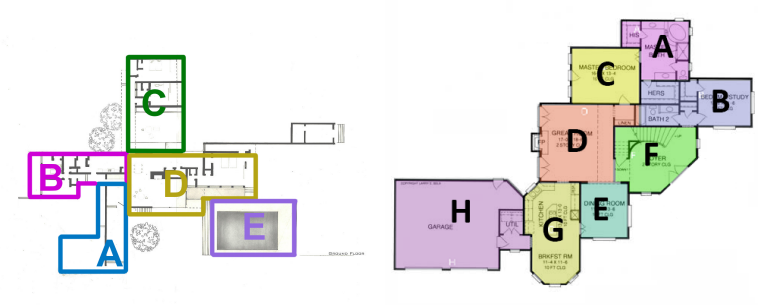


Figure 5.21: Reference layouts.

qualities of the reference layouts with a slightly different arrangement: this would allow the user to explore variations of an optimal solution. Figure 5.23 depicts the graphs of quality improvement with increasing generations, left of the first experiment and right of the second.
As mentioned before, designers usually take advantage of layouts that they designed in the past as references to create a layout for the upcoming project. Both experiments show that ILRS can assist this way the designer. With this type of input, the system can take more accurate decisions in the early stages of the design.

5.8.4 Detailed analysis of problem formulation

In this subsection, we analyse in detail the key aspects of the system to provide a more clear understanding of the strategies applied in problem formulation and the

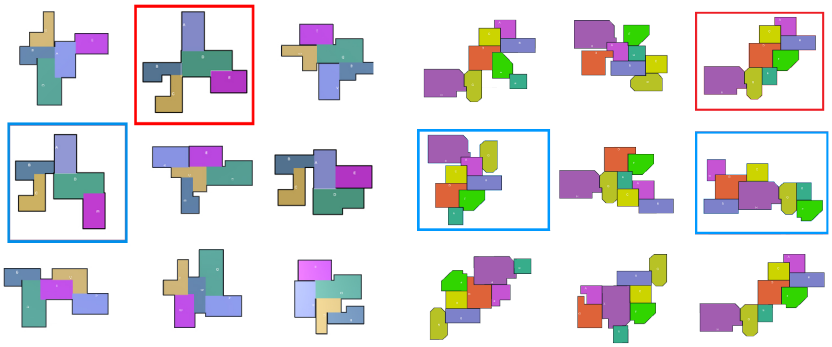


Figure 5.22: Screenshots of the last generations of the two validation experiments

reasons behind them. The analysis is conducted through explaining the differences in the problem formulation of our method and one of the closest systems (EPSAP).

We start discussing the representation of the solution space, which is one of the most important elements of an optimization process. We deal with a space of layouts, and the space units shape and their arrangement are the most substantial factors that lead to that space. An ideal representation should be able to address all possible shapes and arrangements. Table 5.3 shows that the four systems support, to some extent, the generation of non-rectangular shapes; however, only our approach is able to generate arbitrary polygons. The ability of a solution space representation to address different arrangements depends on the type of rigid transformations that are defined in the optimization method. EPSAP [Rodrigues et al., 2013a] is the approach which so far covered the widest range of transformations; however, only 90 degree rotations are supported in this method. In our approach, the orientation of a space unit is defined based on the angle of the selected edges when the attaching operator is performed, and as arbitrary polygons are supported, edges angles can be varied. Therefore, it can be concluded that due to the absence of any angle restriction, ILRS supports a wider range of arrangements.

The main goal of Evolutionary Algorithms is evolving the initial population until reaching the set of optimal solutions. The main goal of the SHC stage in EPSAP is to avoid the creation of very low quality layouts at each generation. However, feeding the population with new random solutions does not seem reasonable, due to brute force nature of presented random operations, since new solutions may not have the similar characteristics to the high quality solutions in the current generation. This would result in a lower convergence rate of the algorithm, and this stage seems to be very time consuming. In fact, lacking suitable heuristics may result in performing operations that do not increase layout fitness. For instance, the trans-

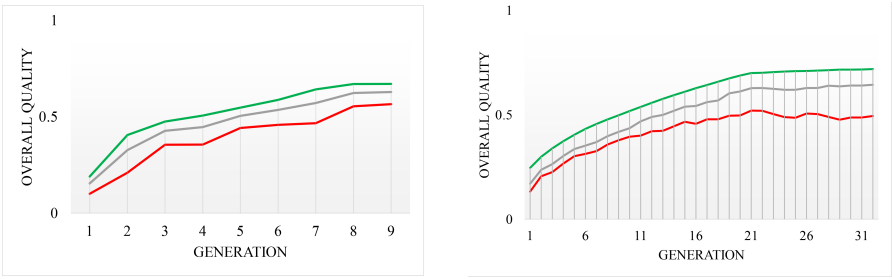


Figure 5.23: The overall quality of two validation experiments. Red is the highest quality layout, green is the lowest quality layout and gray is the the average quality of that generation.

lation operator is applied to make two rectangles closer or hopefully snap them on their edges. The operator uses the distance between their centers multiplied by a Gaussian random value as the length of the translation. However, this likely causes overlap of rectangles if the operator does not consider the orientation of the rectangles as well as the distance between their centers. Furthermore, operations are applied in a cyclical fashion, in a way that all individuals are subjected to a particular operation each cycle. The set of operations in each cycle is independent of previous cycles, therefore operations that do not improve the quality of the individuals may be repeated several times. In contrast, our approach not only applies more accurate attachment operators but it also attempts to avoid repeating operations which do not increase fitness. Our attachment operator considers the distance between the attachment points, instead of the centers of polygons, and the angle between edges resulting in a transformation that never generates overlap in the case of attaching two convex polygons. In order to decrease the chance of invalid attachments for irregular polygons, the history of invalid attachments in previous generations is stored with intention of avoiding invalid attachments in following generations.

5.9 Discussion and Limitations

The previous section has demonstrated that our system is not only able to provide layouts with similar characteristics to human made floor plans, in a time much shorter than that usually employed by an architect, but it also offers some advantages over relevant systems: it supports a wider range of inputs, the solution space is more comprehensive, it supports users interaction, optimizations use more architectural qualities, and it performs the search process in a more efficient way.

It is relevant to discuss more in detail the meaning of some parameters. Let us start by means of an illustrative example. Take, for instance, a hospital: the physical relationships or connections between space units play an essential role in transferring patients and medical instruments. Therefore, the designer may give a higher weight to circulation quality. Hospitals, usually, are built at a site which is larger than the building area, and the designer has more freedom to find an optimal arrangement provided it does not exceed the site boundary. Therefore, the compactness and overflow qualities would get lower weights in the optimization process. Privacy is more important in some spaces than in others, and this would force the designer to setup the openings, especially windows, very precisely. In contrast, when designing the layout of a prison the room visibility dimension, which is somehow the inverse or complement of visual privacy, should have higher value than other quality metrics such as circulation. Thus, in general, input parameters such as connections, or area flexibility and weights of quality metrics in the fitness function define the characteristics of the layout solutions that our system generates.

Given a set of space units, changing the contribution of spatial metrics in the overall quality function can change the whole set of layouts that the user is exposed to in each generation. Besides users preferences, the type of the building should be considered to achieve the target design driven by the appropriate fitness function. ILRS provide a wide flexibility for the designer in terms of shape and arrangement of the layouts. If the user is searching for the best arrangement of a set of known (in terms of geometrical attributes) space units, as is usually the case in procedural generation for video game design, s/he may set the area flexibility of all input space units to zero and leave the topological column as empty. Then, the solution space is reduced to the space of all possible arrangements. On the other hand, if the designer wanted to find the best possible shape of a finite number of space units which satisfy a particular topological matrix, then s/he would give high values to the area flexibility and set the adjacency values to 1 according to the topological matrix. Then the solution space is reduced to layouts which satisfy the topological matrix and the area flexibility thresholds.

Some limitations of our proposal capabilities, which lead to remaining issues that are of interest for future research are:

- One of the main drawbacks of our system is determining the contribution of the user ratings at each generation, as it is not intuitive and is calculated empirically. In (5.9) n defines the slope of a function that sets how fast the user contribution is raised over generations. On one hand, a greater value

of n results in faster convergence; it leads to the creation of layouts similar to user selected items, but which are not necessary high quality in terms of architectural guidelines. On the other hand, a lower value of n causes slower convergence and generating layouts with higher architectural quality but not necessary what the user would expect. Improving this is a research challenge.

- Another limitation of systems such as ours is that they require for the input setting up a large number of constraints. The way of defining them is very unfamiliar when considering what architects are used to work with. Designers (and architects too) normally prefer to deal with more abstract and less complex tools. In future work, we plan to provide the architect/designer with some semantic presets which determine the specific values of the constraints to achieve some predefined layout types such as residential, hospital or school.

- Finally, the layouts in our system are currently generated at the ground level; 2D layouts can be extruded to provide a 3D view of the floor plan to the user, which is supported by our system. A research avenue is to support floorplans with multiple levels (above or under ground) so that designers could handle more complex scenarios.

Chapter 6

CONCLUSION

6.1 Revisiting the motivation, research questions and contributions

The motivation of this thesis comes from the importance and complexity of the creation of realistic 3D buildings in the development process of visual effects and video games. Modeling buildings can be an expensive process when the digital artists have to design large scale scenes, such as cities and rich interiors. The complexity is rooted in two main stages, the construction of 3D mesh models and the visualization or rendering of the generated models. This work is focused on a part of this issue, through the automatic generation of layouts as a subset of building model creation.
Digital artists are not familiar with the architectural design principles, and thus, there is no guarantee that the created virtual buildings can be perceived by the audiences as realistic as a construction or building made by an architect. On the other hand, while involving architects in the production pipeline could be helpful, part of the complexity of the process would remain, since architects are not usually familiar with technical constraints, such as mesh optimization, that are applied by digital artists in the design process.

Turning specifically to the field of architectural design, within the traditional linear approaches the architect does not have enough support to assess the quality of the layouts s/he produces in the design process, and achieving novel and creative design ideas becomes more difficult, especially when dealing with complex projects. Thus, computerized methods have emerged in the form of design assistant tools to facilitate the layout design processes. In particular, in the past decade, AI based systems have been applied in the movies and video game production pipelines to automate the time consuming process of designing large 3D environments through procedural generation of buildings. However, these intelligent tools have not been able yet to address several aspects of the design process, neither in architecture nor in the media and entertainment industry.

The first research question of this thesis, **How can architectural practices or**

rules be formulated into computational metrics that provide accurate and efficient assessment of a layout? addressed one of the complexity issues: assessing quality, which is a substantial element of the overall problem. The second research question, **How can an optimization method be defined so that it generates and finds the optimal layout solutions based on architectural guidelines and user's preferences?** addressed the layout planning problem: the generation, taking into account objective quality measures and subjective preferences.

With respect to the first question, the thesis has provided two new approaches, measuring circulation quality (in chapter 2) and measuring privacy as a complement of visibility (in chapter 3). The circulation quality measures how appropriate a layout is in connecting different space units of a layout which fulfills the project's requirements. The quality of each circulation path between units is measured based on its length, complexity and priority. The visual privacy measures to which extent the visual information is communicated between different space units. In the system which is the culmination of this research, both measures, and several other quality metrics inspired in the previous literature, are employed.

A first approach to provide an answer to the second research question was an optimization method based on binary tree subdivision, discussed in chapter 4, where its capability to generate rectilinear shape layouts is demonstrated and the circulation quality metric is used in the fitness function of the optimization process. A second system, a lot more ambitious in comprehensiveness of the quality issues addressed, the geometries supported, in the optimisation techniques, was presented mainly in chapter 5. We discuss it more in detail next.

The system is based on an evolutionary algorithm (EA), because these types of methods have a similarity with respect to the iterative nature of of the traditional design process and because they have been able to generate novel solutions in problems subject to multiple constraints. The representation of the solution space should be capable of addressing all the solutions that can be generated. We have identified three main types of factors to differentiate between different layout solutions: space unit geometry, arrangement and opening. For instance two layouts with the same arrangement and opening configurations can represent different solutions if at least one space unit represents different geometrical attributes in one of the layouts. The proposed EA-based optimization method, attempts to cover different regions of the solution space through randomizing the layout arrangement, setting random openings, and applying random deformations to space units shape.

The EA receives constraints regarding the desired layout as input. It starts with a semi-random population and attempts to evolve the quality of individuals iteratively through variation and selection operators. The initial population consists of semi-random layouts instead of purely random ones, so that faster convergence to reach optimal solutions is achieved. The semi-random layouts are valid layouts that satisfy a set of hard constraints.
Two variation operators are proposed, recombination and mutation. The recombination operator generates new offspring, children, as a consequence of combining the topological properties of some layouts, their parents. The mutation method is defined as a particular type of recombination when all the individuals of the parent sets are the same. A parent selection method is used to decrease the probability that the algorithm gets stuck on local optima. The method proposed attempts to select sub layout sets, such that their combinations more likely produce higher quality offspring.

The EA is integrated within an Interactive Layout Recommender System (ILRS), which provides a wide range of interactivity, in order to satisfy contextual and subjective aspects of the design. It allows the user to rate the solutions during the evolutionary process. The user ratings set is applied in the following generations as an indicator to measure the quality of the layouts. Let us recall that other quality metrics included in the fitness function are circulation, privacy, topological, overflow, and compactness. ILRS allows the 3D visualisation of layouts to better support the interactivity. The final layout can be slightly edited by the user as well.

As a system, we showed that ILRS supports more features when compared to other relevant proposed systems. We also showed that the constraints satisfaction, and the interactivity are used in our approach in a way such that we get higher quality layouts at each generation. We validated the system against some near-real world situations; and we ran the system in different scenarios, to estimate its performance. ILRS can handle arbitrary shape polygons and reaches optimal solutions in a reasonable number of generations.

In summary, with ILRS we provided a system which answers the second research question, and improves the state of the art in several ways.

6.2 Limitations and Future Work

In this section we discuss some limitations of our approach, which lead to formulate some issues where further research is needed.

The interactivity, the richness of user intervention, is a key ingredient of our approach, which provides important advantages, as we have discussed. However, it is an initial formulation and it is important to take further steps:

- The user ratings of the layouts at each generation contribute to the measure of quality of the layouts through filtering those of the next generation in terms of fitness. We proposed in chapter 5 a formula, 5.9, where n defines the slope of a function that sets how fast the user contribution is raised over generations. We think this is intuitively correct: we expect the user to understand better the appropriateness of the solutions when the design is more evolved, and this should lead to an increased relative weight of the user ratings in the fitness function. A larger n results in a faster convergence; it leads to the creation of layouts similar to user selected items, but which are not necessary high quality in architectural terms. A smaller n leads to slower convergence and to generate layouts with higher architectural quality but not necessary what the user would expect. Improving the weighting of the user input is a research challenge. Perhaps the weighting could be learned through running an extensive number of experiments.

- In our proposal, the weight given to different quality metrics and the input parameters determine the overall characteristics of the layout solutions. Indeed, it should be like that, as the intended building functionalities highly affect its layout. As Hardy and Lammers say: *A functional design can promote skill, economy, conveniences, and comforts; a non-functional design can impede activities of all types, detract from quality of care, and raise costs to intolerable levels* [Hardy and Lammers, 1977].
  However, the input configuration requires setting up a large number of constraints. Furthermore, the way of defining them is very unfamiliar when considering what architects are used to work with. Designers as well as architects normally prefer to deal with more abstract and less complex tools. In future work, we plan to provide the architect/designer with some semantic presets, a sort of templates, which determine the specific values of the constraints. Some examples of predefined layout types could be residential, hospital or school, which would be oriented to achieve layouts of those types. We would need to understand whether these presets are sufficient to lead to appropriate outcomes, or whether more precise input, for instance, through specifying more these templates, is necessary.
  In the table 6.1, we provide examples of some hypotheses we have made about the parameters for the quality metrics according to the design constraints of each building type. The information in the description column

has been gathered from [National Institute of Building Sciences, 2016] and can be used as a guideline to determine the contribution of different quality factors.

- Another important aspect of these type of systems is related to the architectural rules which are used to define quality metrics, in turn included in fitness functions. Some future work along this is:

  - Although we introduced two novel quality metrics, and used several more inspired upon previous research, still there are presented some important quality factors that have not been addressed yet. Some examples are:
  - **Lighting.** Another building quality is related to lighting, related to the artificial and natural amount and quality of the light it receives in different hours of the day. The designer may change the orientation of some space units or even the entire layout to receive more or better light during the day.
    Energy. The arrangement and shape of space units impacts on the transfer of energy between different space units. The quality of a building can also be measured based on its flexibility related to the energy transfer between rooms, or with the environment.

- In our approach, the similarity between the deformed shaped and the original one is measured based on the difference in area. This is a very simple approach, and the similarity measurement could be more accurate by involving geometrical characteristics of the shapes in the similarity computations. Shape similarity algorithms that are used in computer vision can be appropriate candidates to calculate the closeness level of deformed space units to what the user has designed as the original version of the space units.

- Capturing better the semantic structure of the architectural design workflows is another important issue. For instance, usually, architects would prefer to arrange the kitchen next to the living room rather than to the bedroom. Another example is that the area of the living room has a direct relationship to the number of people who are going to live in the space whose layout is being designed; this number can be estimated by the number of bedrooms. Therefore, the living room area is correlated to the number of bedrooms. According to [Merrell et al., 2010] there is a number of these relationships, which architects implicitly apply when designing, based on their past experiences. Moreover, as mentioned in chapter 1, even for an skilled architect,

it is impossible to consider a large quantity of rules in the decision making process, which needs support from automatic tools. Data driven approaches seem to be appropriate to automate the process capturing and inferring semantic relationships in architectural programs. Therefore, another research avenue is to train a system on large datasets of building layouts. How to do this, and whether specialised training is needed depending on the project goal, for instance whether a hospital, a school, etc. is targeted are probably issues to be considering when addressing this research.

- Finally, the layouts in our system are currently generated at the ground level; and 2D layouts can be extruded to provide a 3D view of the floor plan to the user, to his/her advantage. A more complex design scenario worth of further research is that of defining appropriate layouts for floorplans with multiple levels, both above and under the ground.

Building Type	**Description**	**Quality Order**
Educational	In the past few decades, educational layouts are becoming increasingly specialized based on educational content type and students' age. Usually, educational buildings host a large number of people, and the architect should design the corridors so that the flow of students is smooth when they are leaving/entering the classes. Moreover, a compact arrangement of rooms should be compatible with absence of leaking between adjacent rooms, and both visual and vocal privacy should be taken into account carefully in the design process.	$Pr \simeq Cr > Cp \simeq OV$
Healthcare (Hospital)	Hospitals are among the most complex building types, as they comprise different services. A hospital layout consists of different functional units and each unit needs to fulfil a different set of constraints. In general, the physical relationships between space units is the key element in defining the configuration of the hospital. Moreover, since hospitals host different kinds of patients, and not all the patients get treated in the same space unit, separation and privacy play have a considerable contribution in the design process	$Pr > Cr > Cp > OV$

Jail	A jail should be oppressive and give the prison officers an extensive view to monitor prisoners. Moreover, the privacy of prisoners has the least importance among other factors.	$Cr > Cp > Ov > Pr$
Office	The office building should offer the staff increased satisfaction, productivity and flexibility. On the other hand since offices are usually built in commercial centres and allocating space is costly in such environments, the owner more likely expects the layout to be as compact as possible.	$Pr > Cr > Ov \simeq Cp$
Parking	The current increasing number of automobiles around the world requires well designed parking spaces in close proximity to destinations. The movement pattern of cars, as the most important design element, can be highly affected by the boundary shape of the layout. Therefore, overflow and circulation are more important than the rest of quality metrics.	$Cr > OV > Cp > Pr$

Table 6.1: A basic guideline to set quality constraints

Bibliography

[Acquisti et al., 2015] Acquisti, A., Brandimarte, L., and Loewenstein, G. (2015). Privacy and human behavior in the age of information. *Science*, 347(6221):509–514.

[Afyouni et al., 2012] Afyouni, I., Ray, C., and Claramunt, C. (2012). Spatial models for context-aware indoor navigation systems: A survey. *Journal of Spatial Information Science*, (4).

[Allison et al., 2010] Allison, C., Miller, A., Sturgeon, T., Nicoll, J. R., and Perera, I. (2010). Educationally enhanced virtual worlds. In *Frontiers in Education Conference (FIE), 2010 IEEE*, pages T4F–1–T4F–6.

[Amanatides, 1987] Amanatides, J. (1987). Realism in computer graphics: A survey. *IEEE Computer Graphics and Applications*, (1):44–56.

[Ansary and Shalaby, 2014] Ansary, A. M. E. and Shalaby, M. F. (2014). Evolutionary optimization technique for site layout planning. *Sustainable Cities and Society*, 11:48–55.

[Araújo et al., 2009] Araújo, G. A., Carvalho Jr, F. H., and Corrêa, R. C. (2009). Implementing Endogenous and Exogenous Connectors with the Common Component Architecture. In *Proceedings of the 2009 Workshop on Component-Based High Performance Computing*, CBHPC '09, pages 12:1—-12:4, New York, NY, USA. ACM.

[Arvin and House, 2002] Arvin, S. A. and House, D. H. (2002). Modeling architectural design objectives in physically based space planning. *Automation in Construction*, 11(2):213–225.

[Bahrehmand et al., 2014a] Bahrehmand, A., Evans, A., and Blat, J. (2014a). A Computational Metric of the Quality of Circulation in Interior Spaces. In *The International Conference on Information Visualization Theory*, page 8, Lisbon.

[Bahrehmand et al., 2014b] Bahrehmand, A., Evans, A., and Blat, J. (2014b). Recommendation of Floor Plan Layouts Based on Binary Trees. In *egice*, pages 1–10, Cardiff.

[Bao et al., 2013] Bao, F., Yan, D.-M., Mitra, N. J., and Wonka, P. (2013). Generating and exploring good building layouts. *ACM Transactions on Graphics (TOG)*, 32(4):122.

[Bärtschi and Suri, 2014] Bärtschi, A. and Suri, S. (2014). Conflict-free chromatic art gallery coverage. *Algorithmica*, 68(1):265–283.

[Benedikt, 1979] Benedikt, M. (1979). To take hold of space: isovists and isovist fields. *Environment and Planning B*.

[Bentley and Wakefield, 1997] Bentley, P. J. and Wakefield, J. P. (1997). Conceptual evolutionary design by a genetic algorithm. *Engineering design and automation*, 3:119–132.

[Bhatt et al., 2013] Bhatt, M., Borrmann, A., Amor, R., and Beetz, J. (2013). Architecture, computing, and design assistance. *Automation in Construction*, (32):161–164.

[Bittermann et al., 2006] Bittermann, M., Sariyildiz, S., Ciftcioglu, Ö., and Others (2006). Visual space perception model identification by evolutionary search. In *DS 36: Proceedings DESIGN 2006, the 9th International Design Conference, Dubrovnik, Croatia*.

[Bittermann and Ciftcioglu, 2008] Bittermann, M. S. and Ciftcioglu, O. (2008). Visual perception model for architectural design. *Journal of Design Research*, 7(1):35–60.

[Bittner and Wonka, 2003] Bittner, J. and Wonka, P. (2003). Visibility in computer graphics. *Environment and Planning B: Planning and Design*, 30(5):729–755.

[Boddy and Dean, 1994] Boddy, M. and Dean, T. L. (1994). Deliberation scheduling for problem solving in time-constrained environments. *Artificial Intelligence*, 67(2):245–285.

[Borrmann et al., 2013] Borrmann, D., De Rezende, P. J., De Souza, C. C., Fekete, S. P., Friedrichs, S., Kröller, A., Nüchter, A., Schmidt, C., and Tozoni, D. C. (2013). Point guards and point clouds: Solving general art gallery problems. In *Proceedings of the twenty-ninth annual symposium on Computational geometry*, pages 347–348. ACM.

[Brabazon et al., 2015] Brabazon, A., O'Neill, M., and McGarraghy, S. (2015). Introduction to Evolutionary Computing. In *Natural Computing Algorithms*, pages 17–20. Springer.

[Bustos et al., 2007] Bustos, B., Keim, D., Saupe, D., and Schreck, T. (2007). Content-based 3D object retrieval. *Computer Graphics and Applications, IEEE*, 27(4):22–27.

[Cardamone et al., 2011] Cardamone, L., Loiacono, D., and Lanzi, P. L. (2011). Interactive evolution for the procedural generation of tracks in a high-end racing game. In *Proceedings of the 13th annual conference on Genetic and evolutionary computation*, pages 395–402. ACM.

[Chatham, 2014] Chatham, B. (2014). Chatham Design Group.

[Chen and Chang, 2006] Chen, T.-C. and Chang, Y.-W. (2006). Modern floorplanning based on B*-tree and fast simulated annealing. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 25(4):637–650.

[Choset, 1997] Choset, H. (1997). Incremental construction of the generalized voronoi diagram, the generalized voronoi graph, and the hierarchical generalized voronoi graph. In *Proc. First CGC Workshop on Computational Geometry*.

[Choset and Burdick, 2000] Choset, H. and Burdick, J. (2000). Sensor Based Exploration: The Hierarchical Generalized Voronoi Graph.

[Chu and Young, 2004] Chu, C. C. N. and Young, E. F. Y. (2004). Nonrectangular shaping and sizing of soft modules for floorplan-design improvement. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 23(1):71–79.

[Church and Marston, 2003] Church, R. L. and Marston, J. R. (2003). Measuring Accessibility for People with a Disability. *Geographical Analysis*, 35(1):83–96.

[Ciftcioglu et al., 2006] Ciftcioglu, Ö., Bittermann, M. S., and Sariyildiz, I. S. (2006). Towards computer-based perception by modeling visual perception: a probabilistic theory. In *Systems, Man and Cybernetics, 2006. SMC'06. IEEE International Conference on*, volume 6, pages 5152–5159. IEEE.

[Coates et al., 2005] Coates, P., Derix, C., Krakhofer, I. S. P., and Karanouh, A. (2005). Generating architectural spatial configurations. Two approaches using Voronoi tessellations and particle systems.

[Dachsbacher et al., 2007] Dachsbacher, C., Stamminger, M., Drettakis, G., and Durand, F. (2007). Implicit visibility and antiradiance for interactive global illumination. In *ACM Transactions on Graphics (TOG)*, volume 26, page 61. ACM.

[Demirbas and Demirkan, 2000] Demirbas, O. O. and Demirkan, H. (2000). Privacy dimensions: A case study in the interior architecture design studio. *Journal of Environmental Psychology*, 20(1):53–64.

[Dik-Lun et al., 2004] Dik-Lun, H. H., Hu, H., and Lee, D.-l. (2004). Semantic Location Modeling for Location Navigation in Mobile Environment. In *In Proc. Of the IEEE International Conference on Mobile Data Management (MDM*, pages 52–61.

[Durand, 2000] Durand, F. (2000). A multidisciplinary survey of visibility. *ACM SIGGRAPH Course Notes: Visibility, Problems, Techniques, and Applications*.

[Dürr and Rothermel, 2003] Dürr, F. and Rothermel, K. (2003). On a Location Model for Fine-Grained Geocast. In Dey, A., Schmidt, A., and McCarthy, J., editors, *UbiComp 2003: Ubiquitous Computing SE - 2*, volume 2864, pages 18–35. Springer Berlin Heidelberg.

[Eggert et al., 1992] Eggert, D. W., Bowyer, K. W., and Dyer, C. R. (1992). Aspect graphs: State-of-the-art and applications in digital photogrammetry. In *Proc. ISPRS 17th Cong.: Int. Archives Photogrammetry Remote Sensing*, pages 633–645.

[Eiben and Smith, 2003] Eiben, A. E. and Smith, J. E. (2003). *Introduction to evolutionary computing*. Springer Science & Business Media.

[El Ansary and Shalaby, 2014] El Ansary, A. M. and Shalaby, M. F. (2014). Evolutionary optimization technique for site layout planning. *Sustainable Cities and Society*, 11:48–55.

[Elezkurtaj and Franck, 2001] Elezkurtaj, T. and Franck, G. (2001). *Evolutionary algorithms in urban planning*. na.

[Engelhardt, 2013] Engelhardt, T. (2013). *Efficient From-Point Visibility for Global Illumination in Virtual Scenes with Participating Media*. PhD thesis, Karlsruhe, Karlsruher Institut f{ü}r Technologie (KIT), Diss., 2013.

[Fisher et al., 2011] Fisher, M., Savva, M., and Hanrahan, P. (2011). Characterizing structural relationships in scenes using graph kernels. In *ACM Transactions on Graphics (TOG)*, volume 30, page 34. ACM.

[Fisher-Gewirtzman and Wagner, 2003] Fisher-Gewirtzman, D. and Wagner, I. A. (2003). Spatial openness as a practical metric for evaluating built-up environments. *Environment and Planning B: Planning and Design*, 30(1):37–49.

[Fuchs et al., 1980] Fuchs, H., Kedem, Z. M., and Naylor, B. F. (1980). On visible surface generation by a priori tree structures. In *Computer Graphics*, pages 124–133.

[Fujiyoshi and Murata, 2000] Fujiyoshi, K. and Murata, H. (2000). Arbitrary convex and concave rectilinear block packing using sequence-pair. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 19(2):224–233.

[Georgiou, 2006] Georgiou, M. (2006). *Architectural privacy: A topological approach to relational design problems*. PhD thesis, UCL (University College London).

[Gouraud, 1971] Gouraud, H. (1971). Computer display of curved surfaces. Technical report, DTIC Document.

[Greuter et al., 2003] Greuter, S., Parker, J., Stewart, N., and Leach, G. (2003). Real-time procedural generation ofpseudo infinite'cities. In *Proceedings of the 1st international conference on Computer graphics and interactive techniques in Australasia and South East Asia*, pages 87—-ff. ACM.

[Guthe and Klein, 2003] Guthe, M. and Klein, R. (2003). Automatic texture atlas generation from trimmed NURBS models. In *Computer Graphics Forum*, volume 22, pages 453–461. Wiley Online Library.

[Hahn et al., 2006] Hahn, E., Bose, P., and Whitehead, A. (2006). Persistent real-time building interior generation. In *Proceedings of the 2006 ACM SIGGRAPH symposium on Videogames*, pages 179–186. ACM.

[Hanley Wood, 2007] Hanley Wood (2007). *Essential House Plan Collection: 1500 Best Selling Home Plans*.

[Harding and Derix, 2011] Harding, J. and Derix, C. (2011). Associative Spatial Networks in Architectural Design: Artificial Cognition of Space using Neural Networks with Spectral Graph Theory. In *Design Computing and Cognition'10*, pages 305–323. Springer.

[Hardy and Lammers, 1977] Hardy, O. B. and Lammers, L. P. (1977). *Hospitals, the planning and design process*. Aspen Publishers.

[Hasenfratz et al., 2003] Hasenfratz, J.-M., Lapierre, M., Holzschuch, N., and Sillion, F. (2003). A Survey of Real-time Soft Shadows Algorithms. In *Computer Graphics Forum*, volume 22, pages 753–774. Wiley Online Library.

[Hastings et al., 2009] Hastings, E. J., Guha, R. K., and Stanley, K. O. (2009). Evolving content in the galactic arms race video game. In *Computational Intelligence and Games, 2009. CIG 2009. IEEE Symposium on*, pages 241–248. IEEE.

[Homayouni, 2000] Homayouni, H. (2000). A Survey of Computational Approaches to Space Layout Planning (1965-2000). *Department of Architecture and Urban Planning University of Washington*.

[Homayouni, 2007] Homayouni, H. (2007). *A genetic algorithm approach to space layout planning optimization*. PhD thesis, University of Washington.

[house designers., 2016] house designers., T. (2016). thehousedesigners.

[Indraprastha and Shinozaki, 2012a] Indraprastha, A. and Shinozaki, M. (2012a). Computational models for measuring spatial quality of interior design in virtual environment. *Building and Environment*, 49:67–85.

[Indraprastha and Shinozaki, 2012b] Indraprastha, A. and Shinozaki, M. (2012b). Computational models for measuring spatial quality of interior design in virtual environment. *Building and Environment*, 49:67–85.

[Karlen, 2011] Karlen, M. (2011). THE FIRST PLANNING STEPS. In *Space Planning Basics*, page 28. Wiley.

[Kelly and McCabe, 2006] Kelly, G. and McCabe, H. (2006). A survey of procedural techniques for city generation. *ITB Journal*, 14:87–130.

[Key et al., 2008] Key, S., Grosss, M., and Do, E. (2008). Computing Spatial Qualities For Architecture. *Proceeding of ACADIA*.

[Knecht and Koenig, 2010] Knecht, K. and Koenig, R. (2010). Generating floor plan layouts with kd trees and evolutionary algorithms. In *Generative Art Conf*, pages 238–253.

[Koenig and Knecht, 2014] Koenig, R. and Knecht, K. (2014). Comparing two evolutionary algorithm based methods for layout generation: Dense packing versus subdivision. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, 28(03):285–299.

[Koenig and Schneider, 2012] Koenig, R. and Schneider, S. (2012). Hierarchical structuring of layout problems in an interactive evolutionary layout system. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, 26(02):129–142.

[Kwon et al., 2009] Kwon, D. Y., Gross, M. D., and Yi-Luen Do, E. (2009). ArchiDNA: An interactive system for creating 2D and 3D conceptual drawings in architectural design. *Computer-Aided Design*, 41(3):159–172.

[Leblanc et al., 2011] Leblanc, L., Houle, J., and Poulin, P. (2011). Component-based modeling of complete buildings. In *Proceedings of Graphics Interface 2011*, pages 87–94. Canadian Human-Computer Communications Society.

[Lee et al., 2008] Lee, J., Eastman, C., Lee, J., Jeong, Y., and Kannala, M. (2008). Accessible Distance Measurement Using GT Metric Graph. In *CIB W084,Georgia Institute of Technology*, pages 15–16.

[Lévy et al., 2002] Lévy, B., Petitjean, S., Ray, N., and Maillot, J. (2002). Least squares conformal maps for automatic texture atlas generation. *ACM Transactions on Graphics (TOG)*, 21(3):362–371.

[Little et al., 2005] Little, L., Briggs, P., and Coventry, L. (2005). Public space systems: Designing for privacy? *International journal of human-computer studies*, 63(1):254–268.

[Liu and Pang, 2009] Liu, N. and Pang, M.-Y. (2009). A survey of shadow rendering algorithms: projection shadows and shadow volumes. In *Computer Science and Engineering, 2009. WCSE'09. Second International Workshop on*, volume 1, pages 488–492. IEEE.

[Lobos and Donath, 2010] Lobos, D. and Donath, D. (2010). The problem of space layout in architecture: A survey and reflections.

[Lodi et al., 2002] Lodi, A., Martello, S., and Monaci, M. (2002). Two-dimensional packing problems: A survey. *European journal of operational research*, 141(2):241–252.

[López-Camacho et al., 2013] López-Camacho, E., Ochoa, G., Terashima-Marín, H., and Burke, E. K. (2013). An effective heuristic for the two-dimensional irregular bin packing problem. *Annals of Operations Research*, 206(1):241–264.

[Ma et al., 2014] Ma, C., Vining, N., Lefebvre, S., and Sheffer, A. (2014). Game level layout from design specification. In *Computer Graphics Forum*, volume 33, pages 95–104. Wiley Online Library.

[Maaroju, 2009] Maaroju, N. (2009). *Choosing the Best Heuristic for a NP-Problem*. PhD thesis, THAPAR UNIVERSITY, PATIALA.

[Marson and Musse, 2010] Marson, F. and Musse, S. R. (2010). Automatic Real-time Generation of Floor Plans Based on Squarified Treemaps Algorithm. *Int. J. Comput. Games Technol.*, 2010:7:1—-7:10.

[Martínez and Yannakakis, 2011] Martínez, H. P. and Yannakakis, G. N. (2011). Mining multimodal sequential patterns: a case study on affect detection. In *Proceedings of the 13th international conference on multimodal interfaces*, pages 3–10. ACM.

[McKenna, 1987] McKenna, M. (1987). Worst-case optimal hidden-surface removal. *ACM Transactions on Graphics (TOG)*, 6(1):19–28.

[Merrell and Schkufza, 2008] Merrell, P. and Schkufza, E. (2008). Interactive Furniture Layout Using Interior Design Guidelines.

[Merrell et al., 2010] Merrell, P., Schkufza, E., and Koltun, V. (2010). Computer-generated residential building layouts. In *ACM SIGGRAPH Asia 2010 papers*, SIGGRAPH ASIA '10, pages 181:1—-181:12, New York, NY, USA. ACM.

[Merrell et al., 2011] Merrell, P., Schkufza, E., Li, Z., Agrawala, M., and Koltun, V. (2011). Interactive furniture layout using interior design guidelines. In *ACM SIGGRAPH 2011 papers*, SIGGRAPH '11, pages 87:1—-87:10, New York, NY, USA. ACM.

[Moravec and Elfes, 1985] Moravec, H. P. and Elfes, A. (1985). High resolution maps from wide angle sonar. *Robotics and Automation. Proceedings. 1985 IEEE International Conference on*, 2:116–121.

[Müller et al., 2006] Müller, P., Wonka, P., Haegler, S., Ulmer, A., and Van Gool, L. (2006). Procedural modeling of buildings. *Acm Transactions On Graphics (Tog)*, 25(3):614–623.

[Mustafa et al., 2010] Mustafa, F. A., Hassan, A. S., and Baper, S. Y. (2010). Using space syntax analysis in detecting privacy: a comparative study of traditional and modern house layouts in Erbil city, Iraq. *Asian Social Science*, 6(8):157.

[Naseri et al., 2015] Naseri, S., Bahrehmand, A., and Ding, C. (2015). An Improved Collaborative Recommendation System by Integration of Social Tagging Data. In *Recommendation and Search in Social Networks*, pages 119–138. Springer.

[National Institute of Building Sciences, 2016] National Institute of Building Sciences (2016). WBDG.

[O'rourke, 1987] O'rourke, J. (1987). *Art gallery theorems and algorithms*, volume 57. Oxford University Press Oxford.

[Parish and Müller, 2001] Parish, Y. I. H. and Müller, P. (2001). Procedural Modeling of Cities. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '01, pages 301–308, New York, NY, USA. ACM.

[Pedersen, 1979] Pedersen, D. M. (1979). Dimensions of privacy. *Perceptual and Motor Skills*, 48(3c):1291–1297.

[Peng et al., 2014] Peng, C.-H., Yang, Y.-L., and Wonka, P. (2014). Computing layouts with deformable templates. *ACM Transactions on Graphics (TOG)*, 33(4):99.

[Proskurowski, 1980] Proskurowski, A. (1980). On the generation of binary trees. *Journal of the ACM (JACM)*, 27(1):1–2.

[Puusepp, 2011] Puusepp, R. (2011). Generating circulation diagrams for architecture and urban design using multi-agent systems.

[Rafeh and Bahrehmand, 2012] Rafeh, R. and Bahrehmand, a. (2012). An adaptive approach to dealing with unstable behaviour of users in collaborative filtering systems. *Journal of Information Science*, 38(3):205–221.

[Readinger, 2002] Readinger, W. (2002). Representing and partitioning visual space: applying isovist field theory to human perception. *Journal of Vision*.

[Regateiro et al., 2012] Regateiro, F., Bento, J., and Dias, J. (2012). Floor plan design using block algebra and constraint satisfaction. *Advanced Engineering Informatics*, 26(2):361–382.

[Remolina et al., 1999] Remolina, E., Fernandez, J., Kuipers, B., and Gonzalez, J. (1999). Formalizing Regions in the Spatial Semantic Hierarchy: an AH-Graphs implementation approach. In Freksa, C. and Mark, D., editors, *Spatial Information Theory. Cognitive and Computational Foundations of Geographic Information Science SE - 8*, volume 1661 of *Lecture Notes in Computer Science*, pages 109–124. Springer Berlin Heidelberg.

[Remolina and Kuipers, 2004] Remolina, E. and Kuipers, B. (2004). Towards a general theory of topological maps. *Artif. Intell.*, 152(1):47–104.

[Riley and Morris, 1999] Riley, T. and Morris, L. (1999). *The un-private house*. Museum of Modern Art New York.

[Ritchie et al., 2015] Ritchie, D., Mildenhall, B., Goodman, N. D., and Hanrahan, P. (2015). Controlling procedural modeling programs with stochastically-ordered sequential Monte Carlo. *ACM Transactions on Graphics (TOG)*, 34(4):105.

[Rodrigues et al., 2013a] Rodrigues, E., Gaspar, A. R., and Gomes, Á. (2013a). An evolutionary strategy enhanced with a local search technique for the space allocation problem in architecture, Part 1: Methodology. *Computer-Aided Design*, 45(5):887–897.

[Rodrigues et al., 2013b] Rodrigues, E., Gaspar, A. R., and Gomes, Á. (2013b). An evolutionary strategy enhanced with a local search technique for the space allocation problem in architecture Part 2: Validation and performance tests. *Computer-Aided Design*, 45(5):898–910.

[Sait and Youssef, 1999] Sait, S. M. and Youssef, H. (1999). *VLSI physical design automation: theory and practice*, volume 6. World Scientific.

[Sander et al., 2001] Sander, P. V., Snyder, J., Gortler, S. J., and Hoppe, H. (2001). Texture mapping progressive meshes. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 409–416. ACM.

[Scherzer et al., 2011] Scherzer, D., Wimmer, M., and Purgathofer, W. (2011). A Survey of Real-Time Hard Shadow Mapping Methods. In *Computer Graphics Forum*, volume 30, pages 169–186. Wiley Online Library.

[Schiffenbauer, 2001] Schiffenbauer, R. D. (2001). A survey of aspect graphs. *URL citeseer. ist. psu. edu/schiffenbauer01survey. html*.

[Shaker et al., 2013] Shaker, N., Asteriadis, S., Yannakakis, G. N., and Karpouzis, K. (2013). Fusing visual and behavioral cues for modeling user experience in games. *Cybernetics, IEEE Transactions on*, 43(6):1519–1531.

[Simon, 1977] Simon, H. A. (1977). The structure of ill-structured problems. In *Models of discovery*, pages 304–325. Springer.

[Sutherland et al., 1974] Sutherland, I. E., Sproull, R. F., and Schumacker, R. A. (1974). A characterization of ten hidden-surface algorithms. *ACM Computing Surveys (CSUR)*, 6(1):1–55.

[Tang and Wong, 2004] Tang, X. and Wong, M. D. F. (2004). On handling arbitrary rectilinear shape constraint. In *Proceedings of the 2004 Asia and South Pacific Design Automation Conference*, pages 38–41. IEEE Press.

[Togelius et al., 2015] Togelius, J., Shaker, N., and Nelson, M. J. (2015). The search-baesd approach. In Shaker, N., Togelius, J., and Nelson, M. J., editors, *Procedural Content Generation in Games: A Textbook and an Overview of Current Research*, chapter The search. Springer.

[Vasin et al., 2015] Vasin, Y., Osipov, M. P., Muntyan, S. V., and Kustov, E. A. (2015). Procedural modeling and interactive 3D visualization of objects of the internal structure of buildings and facilities. *Pattern Recognition and Image Analysis*, 25(2):278–280.

[Virirakis, 2003] Virirakis, L. (2003). GENETICA: A computer language that supports general formal expression with evolving data structures. *Evolutionary Computation, IEEE Transactions on*, 7(5):456–481.

[Werner et al., 2000] Werner, S., Krieg-Brückner, B., and Herrmann, T. (2000). Modelling Navigational Knowledge by Route Graphs. In *Spatial Cognition II, Integrating Abstract Theories, Empirical Studies, Formal Methods, and Practical Applications*, pages 295–316, London, UK, UK. Springer-Verlag.

[Westin, 1968] Westin, A. F. (1968). Privacy and freedom. *Washington and Lee Law Review*, 25(1):166.

[Wexler et al., 2005] Wexler, D., Gritz, L., Enderton, E., and Rice, J. (2005). GPU-accelerated high-quality hidden surface removal. In *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware*, pages 7–14. ACM.

[Wiener and Franz, 2005] Wiener, J. and Franz, G. (2005). Isovists as a Means to Predict Spatial Experience and Behavior. In Freksa, C., Knauff, M., Krieg-Brückner, B., Nebel, B., and Barkowsky, T., editors, *Spatial Cognition IV. Reasoning, Action, Interaction SE - 3*, volume 3343 of *Lecture Notes in Computer Science*, pages 42–57. Springer Berlin Heidelberg.

[Wineman and Peponis, 2010] Wineman, J. D. and Peponis, J. (2010). Constructing spatial meaning spatial affordances in museum design. *Environment and Behavior*, 42(1):86–109.

[Wong and Liu, 1986] Wong, D. F. and Liu, C. L. (1986). A New Algorithm for Floorplan Design. In *Proceedings of the 23rd ACM/IEEE Design Automation Conference*, DAC '86, pages 101–107, Piscataway, NJ, USA. IEEE Press.

[Wong and Liu, 1989] Wong, D. F. and Liu, C. L. (1989). Floorplan design of VLSI circuits. *Algorithmica*, 4(1-4):263–291.

[Wong and Chan, 2009] Wong, S. S. Y. and Chan, K. C. C. (2009). EvoArch: An evolutionary algorithm for architectural layout design. *Computer-Aided Design*, 41(9):649–667.

[Woo et al., 1990] Woo, A., Poulin, P., and Fournier, A. (1990). A survey of shadow algorithms. *Computer Graphics and Applications, IEEE*, 10(6):13–32.

[Xu et al., 2015] Xu, W., Wang, B., and Yan, D.-M. (2015). Wall grid structure for interior scene synthesis. *Computers & Graphics*, 46:231–243.

[Yannakakis and Hallam, 2007] Yannakakis, G. N. and Hallam, J. (2007). Entertainment modeling in physical play through physiology beyond heart-rate. In *Affective Computing and Intelligent Interaction*, pages 254–265. Springer.

[Yannakakis et al., 2010] Yannakakis, G. N., Mart\'\inez, H. P., and Jhala, A. (2010). Towards affective camera control in games. *User Modeling and User-Adapted Interaction*, 20(4):313–340.

[Yu et al., 2011] Yu, L.-F., Yeung, S.-K., Tang, C.-K., Terzopoulos, D., Chan, T. F., and Osher, S. J. (2011). Make it home: automatic optimization of furniture arrangement. *ACM Trans. Graph.*, 30(4):86:1—-86:12.

Appendix A

APPENDIX: OTHER PAPERS AND RESEARCH AND DEVELOPMENT PROJECTS

Other Papers

This section points to 3 research papers where I have contributed during my PhD period. The first one is a survey on 3D graphics on the Web, where I surveyed a set of applications. The second and third one are papers applying AI related techniques for collaborative filtering. I was carrying out this type of research before starting my PhD in graphics within the UPF, and the papers contain some results continuing that line. I provide the abstract of the papers, and the appropriate reference.

Research Paper A

Title	3D graphics on the web: A survey
Authors	Alun Evans, Marco Romeo, Arash Bahrehmand, Javi Agenjo, Josep Blat
Journal	Computers and Graphics
Year	2014
doi	$10.1016/j.cag.2014.02.002$
Abstract	In recent years, 3D graphics has become an increasingly important part of the multimedia web experience. Following on from the advent of the X3D standard and the definition of a declarative approach to presenting 3D graphics on the web, the rise of WebGL has allowed lower level access to graphics hardware of ever increasing power. In parallel, remote rendering techniques permit streaming of high-quality 3D graphics onto a wide range of devices, and recent years have also seen much research on methods of content delivery for web-based 3D applications. All this development is reflected in the increasing number of application fields for the 3D web. In this paper, we reflect this activity by presenting the first survey of the state of the art in the field. We review every major approach to produce real-time 3D graphics rendering in the browser, briefly summarise the approaches for remote rendering of 3D graphics, before surveying complementary research on data compression methods, and notable application fields. We conclude by assessing the impact and popularity of the 3D web, reviewing the past and looking to the future.

Research Paper B

Title	Enhancing tag-based collaborative filtering via integrated social networking information
Authors	Sogol Naseri, Arash Bahrehmand, Chen Ding, Chi-Hung Chi
Conference	Advances in Social Networks Analysis and Mining (ASONAM), 2013 IEEE/ACM
Year	2013
doi	10.1145/2492517.2492658
Abstract	Recently, researchers have taken tremendous strides in attempting to synthesize conventional social judgments and automated filtering within recommender systems. In this study, we aim to enhance recommendation efficiency via integrating social networking information with traditional recommendation algorithms. To achieve this objective, we first propose a new user similarity metric that not only considers tagging activities of users, but also incorporates their social relationships, such as friendship and membership, in measuring the closeness of two users. Subsequently, we define a new item prediction method which makes use of both user-to-user similarity and item-to-item similarity. Experimental outcomes on Last.fm show some positive results that attest the efficiency of our proposed approach.

Research Paper C

Title	An Improved Collaborative Recommendation System by Integration of Social Tagging Data
Authors	Sogol Naseri, Arash Bahrehmand, Chen Ding
Book	Recommendation and Search in Social Networks
Year	2015
doi	$10.1007/978-3-319-14379-8_7$
Abstract	Recently a lot of research efforts have been spent on building recommender systems by utilizing the abundant online social network data. In this study, we intend to enhance the recommendation accuracy via integrating social networking information with the traditional recommendation algorithms. To achieve this goal, we first propose a new user similarity metric that not only considers tagging activities of users, but also incorporates their social relationships such as friendship and membership, in measuring the closeness of two users. Then we define a new item prediction method which makes use of both user-to-user similarity and item-to-item similarity. Experimental outcomes on Last.fm data produce the positive results that show the accuracy of our proposed approach.

Research and Development Projects

During my PhD time at the UPF, I have worked on several research and development projects, which are computer graphics applications.

Figure A.1: (Top) arrangement of medical furniture in the a hospital room, (Bottom) description of a sample object.

A 3D customised editor

The main goal of this project, commissioned by the company ROCHE, is to provide a set of software tools to:

- Manage (import, export and save) 3D contents within a Unity application.
- Arrange 3D furniture in pre-designed 3D virtual environments.

The 3D contents are medical machines and equipment, and the virtual environments are 3D hospital layouts. The tools were developed within Unity 3D with C#, to support Roche engineers and sales persons in the setup and visualization of 3D scenes containing the medical machines. The generated scenes are used in

clients visits to Roche HQ, conferences, business fairs, etc. In addition, it provides the user an interactive visualizer to evaluate different object arrangements. Currently, the editor exports scenes for Windows desktop systems and tactile screens. Figure A.1 illustrates two examples of the Roche editor.

Virtual Camera (IMPART)

This project, as a sub project of the European Project IMPART (http://impart.upf.edu), is focused on creating a virtual camera to navigate through 3D scenes using mobile devices. 3D scenes are generated based on the point clouds that were scanned or reconstructed corresponding to different scenes in the IMPART project. The virtual camera takes advantage of augmented reality techniques and the device gyroscope to track the position and rotation of camera, respectively. Moreover, a set of User Interface tools is designed allowing the user to generate an animation based on the transformation of the camera in the scene. Figure A.2 shows an screenshot of the implemented application on iPad.

Figure A.2: Using the virtual camera on an iPad to navigate through a 3D scene.

3D visualization of Bilbao Athletic Club Stadium

GTI, in collaboration with the Mobile Media Content company, has developed a 3D visualization tool for the Bilbao Athletic Club. The developed application provides the near-real time simulation of players interactions in the Athletic Stadium through an interactive visualizer. Figure A.2 shows two examples of the software

application. I have contributed in this project as a 3D developer.

Figure A.3: Near-real time simulation of players and ball in the field

Discussion

Although not every part of these projects and research was directly relevant to my thesis research, they helped to complement my education around several main questions:

- *How can we approach multidisciplinary problems?* For instance, doing research in the field of recommendation systems and artificial intelligence taught me how we can incorporate user's opinion in optimization processes to personalize the floor plans.
- *How can we implement a software application that is capable of reflecting research ideas accurately in a near-real time fashion?*Through contributing to the 3D web survey and IMPART projects, I got familiar with bottlenecks in 3D real time applications.
- Moreover, my experience of working in the ROCHE project includes designing a CAD based tool that not only is highly performant but also offers

an easy-to-work user interface to the end user. This experience helped me as well for the ILRS.